Proyecto de Software 2022

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 2022, se trabajará en una aplicación que permita gestionar un club barrial. Este trabajo surge ante el pedido del Club Deportivo Villa Elisa, quien en este año cumple 100 años desde su fundación.

El trabajo se llevará a cabo en dos (2) 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 distinta información a las personas asociadas del club.
Por un lado, se desarrollará una *aplicación privada* que potencialmente proveerá la siguiente funcionalidad:

- Gestión de las personas asociadas del club (con alguna funcionalidad para la importación y exportación de datos).
- Gestión de las disciplinas y/o actividades.
- Gestión de pagos.
- Mantener opciones de configuración del sistema.
- Administración de roles para aplicación privada.

También se desarrollará una *aplicación pública*, que permitirá visualizar la información de interés para la comunidad y personas asociadas al club. A través de esta aplicación se podrá:

- Obtener información para establecer contacto con el club.
- Conocer los precios de las actividades del club y sus disciplinas.
- Tramitar la credencial digital.
- Visualizar estado de cuenta/socio.
- Registrar la realización de un pago de cuota.

Etapa 2

Fecha de entrega: 24/11 a las 23:59

Aplicación Privada

1.9 Carnet Digital

Este módulo permite a un usuario con el rol de **Operador/a** o **Administrador/a**, visualizar o descargar el carnet digital de cada una de las personas asociadas.

Un carnet digital es una representación de la ficha de una persona que incluye información básica como nombre y apellido, foto, Nro. de asociado/a, fecha de emisión,
código de validación único (QR).
El módulo debe brindar **al menos** la siguiente funcionalidad:

- Dada una persona asociada, se le deberá poder generar un carnet digital.
- Visualizar el estado de cuenta societaria.
- Para la generación se deberá adjuntar foto del asociado y el sistema generará un Código Único de Validación en formato QR.
- El QR se debe armar a partir de la URL donde se visualiza la información del carnet digital del usuario en el Portal de Administración, al que solo puede ingresar un operador o administrador (por ejemplo https://backend.asociados/{ID}/carnet).
- El carnet digital deberá poder exportarse a PDF.

1.9.1 API Carnet digital

Se deberá desarrollar **en la aplicación privada** un nuevo servicio público que permita consultar desde la **aplicación pública** toda la información del carnet de socio y el estado de cuenta actual del mismo.

- Documentación del servicio GET /api/me/license

Aplicación Pública

Se deberá desarrollar **otra aplicación** a la cual accederán los y las socio/as donde podrán obtener información de interés sobre el club, como disciplinas habilitadas, actividades, estado de la cuota societaria (carnet digital), etc.

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**.

2.1 Home + Información de contacto

En la vista principal de la aplicación deberá contar con una explicación resumida del servicio que brinda el portal, y una breve explicación de las acciones que puede realizar un visitante en el mismo:

- Visualizar el listado de disciplinas y sus correspondientes horarios, profesores y costo.
- Validar con nro de socio el estado de la cuota societaria (cuotas adeudadas) y de cada una de las actividades a las que se encuentre inscripto.

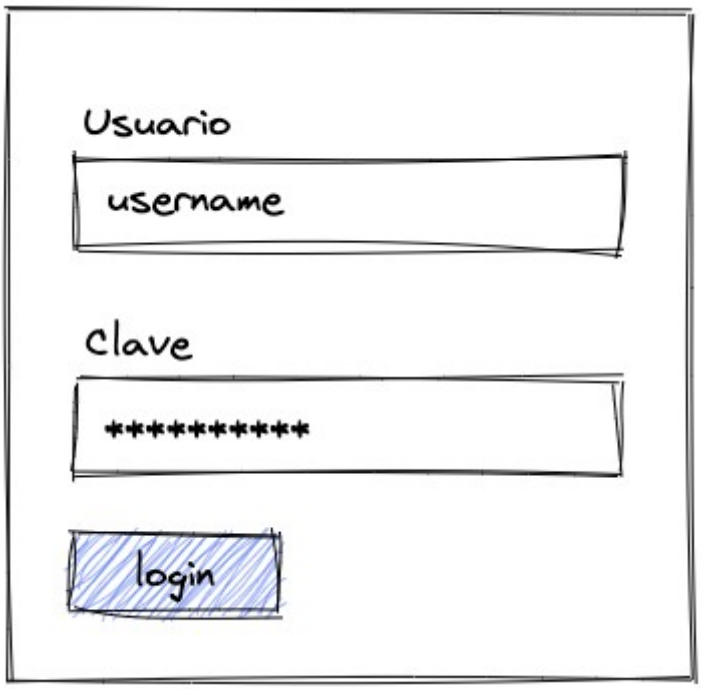
Sumar cualquier otra operación que esté habilitada desde el portal.

2.2 Listado de las Disciplinas

En esta sección se visualizará un listado de las disciplinas que se practican en el club, junto con el día y horario en el que se desarrollan, costos, así como el/la profesor/a. A su vez, el/la socio/a. Para obtener la información necesaria del listado de las disciplinas, deberá consultar la **API de Disciplinas** definida en la *ETAPA 1 - 1.5.2.*

2.3 Autenticación

Implemente un login basado en JSON Web Tokens (JWT), que permita acceder a la información en la aplicación privada. Esta funcionalidad deberá utilizar la **API de Login** definida en 2.3.1.
La persona asociada puede loguearse en la aplicación, para esto es necesario visualizar un formulario de login. El formulario utiliza la API de login JWT mencionada anteriormente.



No existe la funcionalidad de registro de una persona asociada por medio del portal. El alta de las personas asociadas se realiza mediante la aplicación de administración y por un usuario con rol **Administrador/a**.

2.3.1 API de Login

Desarrollar una API que permita autenticar el usuario que ingresa a la aplicación pública.

- Documentación del servicio POST /api/auth

2.4 Pagos

Esta sección permite listar los pagos realizados por el usuario logueado en la aplicación pública, además permite subir el comprobante del pago de la cuota.

2.4.1 Visualización de pagos realizados

Esta sección permite visualizar el listado de cuotas pagas e impagas de la persona asociada que se encuentra logueada en la aplicación pública. Para obtener la información necesaria del listado, deberá consultar la **API de Pagos** definida en la *ETAPA 1 - 1.7.2.*

2.4.2 Realizar pago

Esta sección permite realizar el pago de la cuota a una persona asociada que se encuentra logueada en la aplicación pública.
Lo requerido en este punto es que el asociado pueda subir un comprobante de la cuota pagada por algún medio digital (en formato jpg, pdf, png).
Para realizar esta operación, deberá utilizar la **API de Pagos** definida en la *ETAPA 1 - 1.7.2.*

2.5 Estadísticas

Aquí se visualizará información que pueda ser de utilidad para la administración del club, así como para las personas asociadas. Algunos ejemplos podrían ser: asociados que se encuentran al día, asociados morosos, participación en disciplinas por género/edad etc.
En esta vista se deberán diseñar y desarrollar 3 gráficos distintos. Queda a criterio de las y los desarrolladores/as con una validación previa del ayudante asignado que tipo de gráficos generar y teniendo en cuenta los datos utilizados.

Tener en cuenta que esta funcionalidad se debe realizar en la aplicación *pública*. Se deberá utilizar un componente que sirva para realizar esta implementación.

Consideraciones generales:

- El prototipo debe ser desarrollado utilizando Python, JavaScript, HTML5, CSS3 y PosgreSQL, 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**.
- 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 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 cada una de las API requeridas en el enunciado.
- 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.
- Seleccione una vista (**HTML5** y **CSS3**) para realizar validaciones de 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. 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: Postgres 13
- Intérprete Python: Python 3.10.6
- Node JS: v14.20.0 (npm 6.14.17)
- Servidor web: Nginx 1.18.0-0ubuntu1.3