**Desarrollo del Proyecto**

Contenido:

1. Descripción del análisis de cómo se realizará la implementación. Por ejemplo, creará una base de datos o utilizará los archivos suministrados u otra forma.
2. Descripción de las consultas que se van a implementar
3. Descripción de los archivos generados. Por ejemplo, por programa o por asignatura, por docentes o por otra categoría.
4. Qué método de pandas utilizará para combinar los diferentes archivos.
5. Descripción detallada del desarrollo de la aplicación. Backend y Frontend utilizado.
6. Instalaciones adicionales requeridas.
7. Archivos de la aplicación.
8. Descripción y reporte de anomalías. Por ejemplo, asistentes que no se inscribieron. Asistentes que se hayan validado con correos diferentes.
9. Ajustes realizados a los datos si considera pertinente para presentar informes o consultas solicitadas o propuestas.

Desarrollo:

1. Descripción del análisis de cómo se realizará la implementación. Por ejemplo, creará una base de datos o utilizará los archivos suministrados u otra forma.

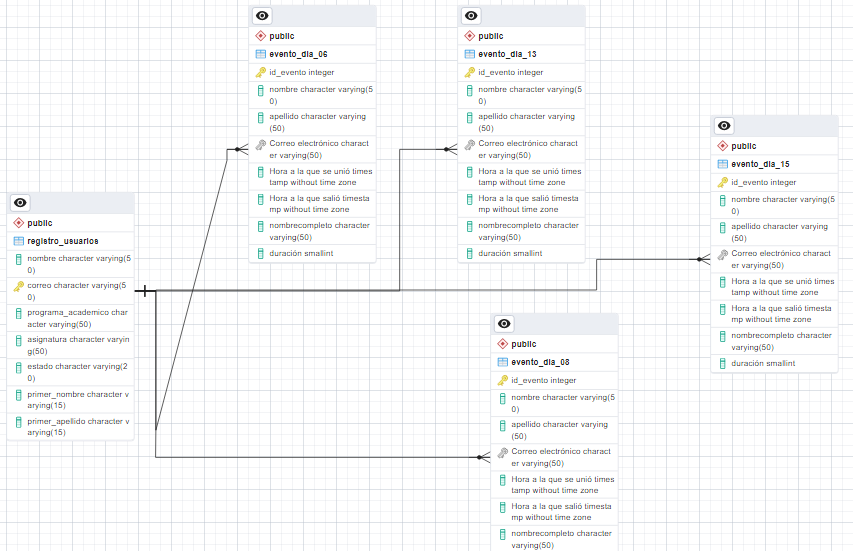
Pasos para realizar la implementación:

1. Se crea un directorio donde se almacena todo el proyecto, para esta ocasión se nombra “Proyecto\_ETL\_Dashboard”
2. Se crean tres directorios:
   1. Directorio para el Backend “Backend”
   2. Directorio para el Frontend “Dashboard”
   3. Directorio para el análisis de datos “ETL”
3. Se tiene como objetivo:
   1. Hacer limpieza de datos
   2. Hacer modelado de base de datos
   3. Hacer diseño de Dashboard
   4. Hacer una API para comunicación entre base de datos y Dashboard
4. La limpieza de datos se trabaja en el directorio de “ETL”, se trabaja con el software libre visual studio code, a traves de archivos jupyter notebook (.ipynb)
5. En el directorio de “ETL” se crean tres directorios:
   1. Directorio de archivos de salida “clean\_data”
   2. Directorio de archivos de entrada “data”
   3. Directorio de archivos de trabajo “src”
   4. Archivos Pipfile (Se creo un ambiente de desarrollo mediante la librería en Python pipenv)
6. En el Directorio “ETL/src” se encuentran tres archivos:
   1. Archivo “limpieza.ipynb” donde procedemos con la limpieza de los datos mediante librerías como pandas y numpy
   2. Archivo “config\_db.py” donde podemos crear la base de datos, crear sus tablas y agregarles la información ya limpia.
      1. En este apartado es importante colocar un archivo “.env” en el directorio “ETL/src” que almacena la clave de PostgreSQL del usuario, de lo contrario tendrá que modificar el archivo “config\_db.py” colocando su clave y demás información que varie con respecto al modelo propuesto como el host, usuario.
      2. Es importante que se descomente el comando que crea la base de datos en el archivo “config\_db.py” el cual dice “# create\_database("eventos")” y se comente todo lo de abajo. Una vez se halla creado se procede a comentar el comando que crea la base de datos y se descomenta lo comentado para ejecución del comando.
   3. Archivo “analisis.ipynb” donde nos conectamos a la base de datos y generamos archivos de salida, generamos funciones que nos permitan realizar gráficos con análisis de valor visual para un usuario final revisando su Dashboard
7. En el directorio “Backend” están los siguientes archivos:
   1. Archivo “main.py” que contendrá el desarrollo de una API REST realizada mediante la librería fastAPI.
      1. Esta se conectará a la base de datos por lo tanto es importante que se tenga un archivo “.env” en el directorio “Backend/” que nos permita acceder a la base de datos mediante su contraseña
   2. Archivo “procesar\_datos.py” que contendrá funciones de valor para el desarrollo de la API, con el fin de evitar aumentar el código y de cierta tener en un archivo todo lo relacionado con la API y en el otro archivo se tienen las funciones de apoyo para el buen funcionamiento del API.
   3. Archivos “Pipfile” que como se ha comentado anteriormente permiten crear un ambiente virtual mediante la librería pipenv y a su vez contiene información de los módulos utilizados para una mayor facilidad en su instalación.
8. En el directorio “Dashboard” están los siguientes archivos:
   1. Archivos de instalación con el framework “Sveltekit”, esta instalación se realiza mediante “npm” en su versión 9.8.1, por lo tanto es importante tener instalado “NodeJS” versión v18.18.0
   2. En la ruta “Dashboard/src” se encuentran archivos de importancia para el desarrollo, estos se encuentran en:
      1. La ruta “Dashboard/src/lib/components” que contiene componentes gráficos para el desarrollo de la aplicación.
         1. Los gráficos se realizan mediante la librería “ChartJS”
      2. La ruta “Dashboard/src/routes/+page.svelte” que contiene la información del diseño del Dashboard y la conexión con el Backend en su totalidad.
   3. En la ruta “Dashboard/node\_modules” se encuentran todos los módulos requeridos con la aplicación, los cuales deben ser instalados para su funcionamiento.
9. Descripción de las consultas que se van a implementar

En este caso nos ubicamos en la ruta “Proyecto\_ETL\_Dashboard/ETL/src/config\_db.py” y se ejecutan las siguientes sentencias SQL para creación y consulta en PostgreSQL:

* Creación de base de datos “CREATE DATABASE eventos”
* Creación de tablas “CREATE TABLE IF NOT EXISTS registo\_usuarios (nombre VARCHAR(50), correo VARCHAR(50) PRIMARY KEY, programa\_academico VARCHAR(50), asignatura VARCHAR(50), estado VARCHAR(20), primer\_nombre VARCHAR(15), primer\_apellido VARCHAR(15))”
* Creación de tablas para cada evento en este caso se hace el ejemplo para el día seis “CREATE TABLE IF NOT EXISTS evento\_06 (ID\_evento serial PRIMARY KEY, Nombre VARCHAR(50), Apellido VARCHAR(50), “Correo electrónico” VARCHAR(50), “Hora a la que se unió” TIMESTAMP, “Hora a la que salió” TIMESTAMP, NombreCompleto VARCHAR(50), duración SMALLINT, FOREIGN KEY (“Correo electrónico”) REFERENCES registro\_usuarios(correo))”
* Inserción de datos, en este punto nos apoyamos en el ORM, para poder insertar datos en ambas tablas

A continuación, se muestra el modelo entidad relación de la base de datos



Las consultas de extracción de los datos cuando nos conectamos a la base de datos se pueden presenciar en la ruta “Proyecto\_ETL\_Dashboard/ETL/src/analisis.ipynb” donde se ejecuta la siguiente sentencia:

* Selección de todos los registros “SELECT \* FROM (nombre de la tabla)”

1. Descripción de los archivos generados. Por ejemplo, por programa o por asignatura, por docentes o por otra categoría.

Los archivos generados además de la base de datos de PostgreSQL, los podemos encontrar en la ruta “Proyecto\_ETL\_Dashboard/ETL/clean\_data/” en este directorio podemos encontrar los siguientes archivos:

* Archivos de limpieza de los datos:
  + El archivo “evento\_dia\_XX.csv” donde XX es el día del evento, se tienen 4 archivos “.csv” de los eventos tratados. Estos archivos tienen la estructura de (Nombre,Apellido,Correo electrónico,Hora a la que se unió,Hora a la que salió,NombreCompleto,duración)
  + El archivo “registro\_usuarios.csv” que tiene la estructura (Nombre,Correo institucional (@uptc.edu.co),Programa Académico,Asignatura,estado,Primer Nombre,Primer Apellido)
  + El archivo “proyecto\_asitencias.xslx” que contiene información en un archivo Excel organizada por programa académico y asignaturas de ese programa académico.

1. Qué método de pandas utilizará para combinar los diferentes archivos.

Primero se procede a realizar una limpieza:

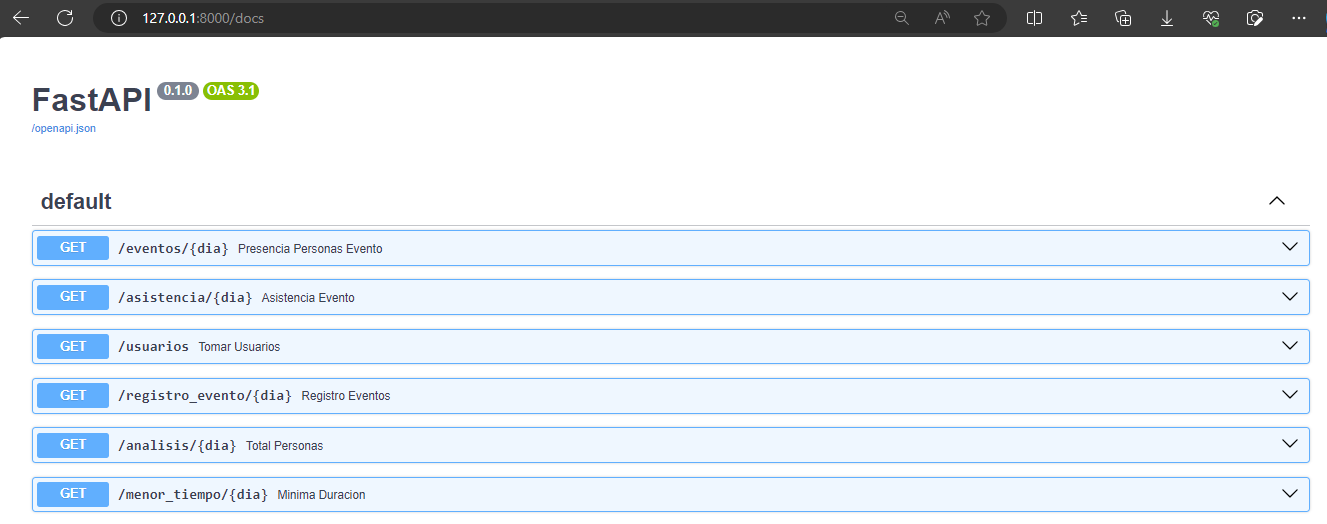
* Registro de usuarios, pasos utilizados con pandas y numpy:
  + Se descompone el correo, se asigna nombre y apellido al campo de nombre donde existan NaNs. (métodos: fillna, apply, join, re)
  + Se transforma la columna Asignatura en mayúsculas y se limpian los espacios a los extremos, luego se procede a reemplazar los valores por los asignados por la ingeniera Jenny, limpiando las diferencias y agrupando coincidencias entre los nombres. (métodos: str, upper, replace)
  + Se procede a transformar la asignatura de los docentes, administrativos y de la gobernación de Boyacá en Profesionales.
  + Se crea una columna de estado que indica cuales personas con asignatura vacía les falta información y cuales están completos
  + Se limpian algunos duplicados y espacios
* Archivos de eventos, pasos utilizados con pandas y numpy:
  + Se crea columna “NombreCompleto” completa combinando nombre y apellido. (métodos: fillna, str, strip)
  + Se arreglan los correos no institucionales Gmail. (métodos: re, str, title)
  + Se arreglan correos no institucionales.
  + Se crea columna de primer nombre y apellido. (métodos: re, str, split, group)
  + Se transforma columna objeto a tiempo. (métodos: to\_datetime, dt, total\_seconds, astype)
  + Duración se crea mediante las nuevas columnas de tiempo, se procede a eliminar columna de duración existente
  + Se generan archivos de salida. (métodos: to\_csv)
* Archivo Excel de salida:
  + Se generan dataframes por cada asignatura de cada programa académico. (métodos: append, isin, unique)
  + Se crea Excel con el método to\_excel usando pd.ExcelWriter

1. Descripción detallada del desarrollo de la aplicación. Backend y Frontend utilizado.

Desarrollo del Backend con el módulo principal fastAPI, para ello nos ubicamos en la ruta “Proyecto\_ETL\_Dashboard/Backend”:

* Tenemos un archivo “procesar\_datos.py” tenemos funciones las cuales son:
  + La función “extraer\_db\_a\_dataframe” crea dataframes de los datos guardados en la base de datos.
  + La función “asistencia\_eventos” crea dataframes de la cantidad de personas que asistieron a cierto evento en cuestión
  + La función “tendencia\_personas” crea dataframes con tiempos, programa académico y la cantidad de personas por ese tiempo y programa académico
  + La función “personas\_por\_programa” nos da un dataframe de programa académico y cantidad de personas por ese evento
  + La función “duracion\_promedio\_sesion” nos da un entero de la duración promedio del evento
  + La función “duración\_std\_sesion” nos da un entero de la desviación estándar
  + La función “total\_personas\_sesion” nos un entero de la suma de las personas en ese evento
  + La función “mínima\_duracion\_sesion” nos da un dataframe con un dato, la persona que menos duro en el evento, otorgando nombre, correo y duración
* En el archivo “main.py” tenemos el desarrollo de la API, donde tenemos que importar el archivo “procesar\_datos.py” y módulos para el desarrollo de la API.
  + Se crean diferentes rutas que se basan en las funciones donde retornan JSON. La documentación con fastAPI se realiza automáticamente al momento de ejecutarla y agregando “/docs” donde de manera detallada se explica la API.
    - Para probar la documentación es necesario ejecutar la api mediante uvicorn main:app –reload y entrando al servidor añades “/docs” al final para poder presenciar la documentación.

A continuación, se muestra una imagen de como se ve la documentación de la API realizada

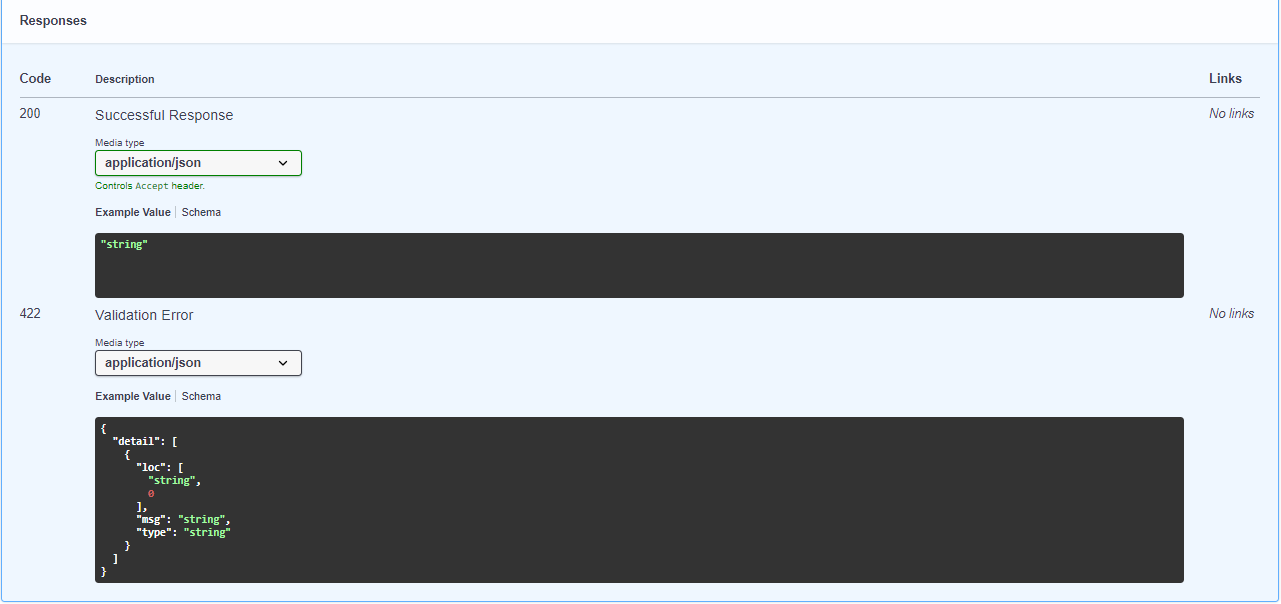


Se ven todas las peticiones posibles con el API, para este caso son de tipo GET. A manera de ejemplo, dando click en la flecha de la derecha del desplegable de la petición /eventos/{día} vemos lo siguiente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Nos indica una breve descripción de lo que realiza, los parámetros que recibe y desplazándonos hacia abajo vemos un ejemplo de las respuestas



Dando click en el botón de try out podemos realizar una prueba sencilla, se procede a dar un ejemplo con la fecha 06 de la siguiente manera

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Lo ejecutamos y muestra lo siguiente:

Forma, Patrón de fondo, Rectángulo

Descripción generada automáticamente

* Tenemos el archivo “Pipfile” donde de igual manera podemos instalar las librerías utilizadas para el funcionamiento

Desarrollo del Frontend con el framework de Sveltekit, para ello nos ubicamos en la ruta “Proyecto\_ETL\_Dashboard/Dashboard”

* Tenemos los componentes para las gráficas “Proyecto\_ETL\_Dashboard/Dashboard/src/lib/components/” realizados con ChartJS, estas contienen el desarrollo de un gráfico de líneas y uno de torta
* Tenemos la aplicación en la ruta “Proyecto\_ETL\_Dashboard/Dashboard/src/routes/+page.svelte” Donde tenemos la conexión a la API desarrollado en el Backend y enviamos esa data a las gráficas y estas a su vez al HTML. Además, cuenta con estilos la aplicación.
* Tener en cuenta que para correr el Frontend se necesita estar corriendo el Backend.

1. Instalaciones adicionales requeridas:

Para el correcto funcionamiento es indispensable que en cada carpeta se ejecuten las instalaciones de las librerías y framework necesarios.

* ETL: Una vez te ubicas en la ruta “Proyecto\_ETL\_Dashboard/ETL” se tiene un archivo Pipfile, para ello mediante la consola en esa ubicación ejecutas pip install pipenv, luego pipenv install, que de manera automática instalará los módulos necesarios y luego con pipenv shell activas el ambiente y listo para la ejecución.
* Backend: De igual manera que en ETL, te ubicas en la ruta “Proyecto\_ETL\_Dashboard/Backend” y repites el proceso del directorio ETL
* Frontend: Te ubicas en la ruta “Proyecto\_ETL\_Dashboard/Dashboard” y si ya tienes node y npm en las versiones ya mencionadas, ejecutas npm install, y de manera automática instala lo que tenga contenido en en el directorio “Proyecto\_ETL\_Dashboard/Dashboard/node\_modules/”

1. Archivos de la aplicación

En este desarrollo se implementa una aplicación web, para su funcionamiento es necesario realizar las instalaciones necesarias de los módulos, una vez hecho se realiza lo siguiente:

* Te ubicas en la ruta “Proyecto\_ETL\_Dashboard/Backend/”, abres la consola y ejecutas uvicorn main:app –reload para ejecutar el servidor en el localhost, por defecto se ejecuta en el puerto 8000
* Te ubicas en la ruta “Proyecto\_ETL\_Dashboard/Dashboard/”, abres la consola y ejecutas npm run dev para ejecutar el servidor en el localhost, en este caso es el puerto 5173, de ejecutarse en otro es importante modificar el Backend en el archivo main.py en la siguiente sentencia “allow\_origins=[<http://localhost:5173>,<http://localhost:1234>] añadiendo a la lista la url que te abrió el servidor del Dashboard.
* Una vez estén los dos corriendo abres el enlace del servidor del Dashboard y listo ya puedes interactuar con el Dashboard.

1. Descripción y reporte de anomalías. Por ejemplo, asistentes que no se inscribieron. Asistentes que se hayan validado con correos diferentes.

En este caso como anomalías solo se trataron los que en estado tenían falta de información como que falta información importante y que es difícil descifrar que dato va en el lugar, Además los usuarios registrados no se trataron ya que por el modelo de base de datos que se relacionan con la tabla principal de registro no se puede añadir usuarios no registrados, por lo tanto, no se encuentran después de la limpieza.

1. Ajustes realizados a los datos si considera pertinente para presentar informes o consultas solicitadas o propuestas

* Revisar los datos de entrada a la base de datos en PostgreSQL en el archivo config\_db.py, si se tienen las mismas configuraciones para localhost, user y ubicar tu contraseña en el archivo “.env” según la ruta que pone en el mismo archivo.
* Revisar los puertos donde se corren los servidores si no tocaría modificar la API en el Backend