

Proyecto → Fase 2

SMART CLASS



Ing. Jesus Guzman, Ing. Alvaro Hernandez, Ing. Luis Espino

Randolph Muy, Jorge Salazar, Josué Pérez, José Véliz

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Estructuras de Datos

Índice

Índice	1
Visión General	3
Objetivos	3
Especificaciones	4
Estudiantes	4
Años	4
Meses	5
Recordatorios	5
Semestres	6
Arquitectura General de Estructuras	7
Estructuras	8
Arbol AVL	8
Listas Enlazadas	8
Matriz Dispersa	8
Arbo B	8
Árbol B de Cursos	8
Cursos del Pensum.	9
Cursos del alumno.	9
Reportes	10
Árbol AVL de Estudiantes	10
Matriz Dispersa de Tareas	10
Lista de tareas	11
Árbol B de Cursos(General)	12
Árbol B de Cursos de un estudiante en un semestre específico	12
Peticiones al servidor	13
Carga masiva	13
Reportes	13
AVL de estudiantes	13
Matriz de Tareas	14
Lista de tareas	14

Árbol B de Cursos(General)	15
Árbol B de Cursos de un estudiante en un semestre específico	15
CRUD Estudiantes	15
Crear un estudiante	15
Modificar estudiante	16
Eliminar un estudiante	17
Obtener información de un estudiante	17
CRUD Recordatorios	17
Crear Recordatorio	17
Modificar Recordatorio	18
Obtener información de un Recordatorio	18
Eliminar Recordatorio	19
CRUD Cursos	19
Crear Curso	19
Cursos de Estudiantes	19
Cursos del Pensum	21
Consideraciones	24



Visión General

A raíz de la situación por la cual atraviesa la sociedad, la digitalización ha sido un total apoyo a casi todas las actividades que se realizan, por lo cual se le solicita a usted como desarrollador, la creación de una aplicación de apoyo a la universidad haciendo uso de los conocimientos y capacidades que ha adquirido hasta el punto en el cual se encuentra en la carrera.

Smart class será una plataforma diseñada para el apoyo a los estudiantes universitarios a organizar de mejor manera las actividades que realizan en su día a día, esto para poder manejar de manera ordenada las mismas, lo cual facilitará la visualización y administración de estas actividades, esta aplicación será capaz de almacenar no solamente las tareas sino también los estudiantes que forman parte de la misma, todos estos datos deben ser analizados y almacenados de manera segura y confiable.

Objetivos

- Que el estudiante se familiarice con el lenguaje de programación Python.
- Que el estudiante sepa involucrarse en el ámbito del manejo de la memoria.
- Familiarizarse con el uso de Git.
- Que el estudiante se familiarice con el manejo de lectura de archivos.
- Que el estudiante se familiarice con el uso de frameworks de desarrollo web.
- Comprender el uso de estructuras de datos no lineales.

Especificaciones

En esta fase se solicita crear una aplicación web mediante un servidor desarrollado en el lenguaje de programación Python apoyado de algún cliente (recomendación Postman o Insomnia) para poder realizar peticiones al mismo, cargando los datos obtenidos de la fase anterior tras pasar por el debido análisis, para analizar los archivos que fueron generados, se recomienda usar un analizador desarrollado en la herramienta Ply para poder obtener los datos que serán cargados a las estructuras de datos que se solicitan y detallan posteriormente en el enunciado del proyecto.

Estudiantes


Los estudiantes que fueron filtrados en la fase anterior, en esta fase se almacenarán en un **árbol AVL**, en el cual el identificador con el que tendrán que ordenarse será el número de **carnet**, a los atributos de los estudiantes se le agrega una estructura que son los años que ha estado activo en el sistema, esta estructura se especifica posteriormente.

Atributos de cada estudiante:

- Número de carnet
- DPI
- Nombre
- Carrera
- Correo
- Password
- Créditos
- Edad
- **Lista de años**

Años

El almacenamiento de años permite a los estudiantes registrar los años que ha cursado o están cursando en la universidad, de esta manera al crear un estudiante será posible tener un conjunto de años asociados, permitiendo registrar de una manera más ordenada el conjunto de cursos y actividades que tiene, para ello se podrá registrar la cantidad de años que se deseen dentro de



una estructura dinámica, cuidando el no repetir un mismo año, ya que esto mostraría una advertencia.

La estructura dinámica que se debe utilizar para almacenar los años que un estudiante desea ingresar se maneja de manera libre, esto quiere decir, que el programador encargado debe utilizar su criterio y elegir la implementación que mejor le parezca para almacenar los años que cada estudiante utiliza, como recomendación se puede utilizar una estructura de datos lineal como la lista doblemente enlazada.

Por último un aspecto que se debe tomar en cuenta es que esta estructura además de almacenar los años, cada nodo debe contener dos apuntadores a dos estructuras, la primera es la estructura de los semestres y la segunda es la estructura de meses.

Meses

La estructura relacionada a los meses debe ser una lista doblemente enlazada, en la cual se podrá almacenar los meses que contengan alguna actividad que el estudiante desea agregar, por esto mismo no será totalmente necesario la existencia de todos los meses en un año, sino en cambio sólo se almacenará los meses en donde existan actividades por realizar.

Algo importante a tener en cuenta con esta estructura es que en cada nodo no solo se debe almacenar la información del mes que corresponda, sino además debe existir un apuntador a una matriz dispersa, de esta manera se permitirá que exista una matriz dispersa por cada mes.

Recordatorios

Los recordatorios que se manejan en esta aplicación son como tal todas las tareas o actividades que los estudiantes desean registrar y manejar, para ello deben utilizar una matriz dispersa que almacenará dentro de su estructura una lista de tareas, correspondientes a la fecha y hora establecida.

La matriz dispersa que se debe realizar contendrá la siguiente distribución:

- ★ **Filas:** Contendrán las distintas horas correspondientes al día, el rango del que puede almacenarse una tarea será dentro de las 24 horas, de tal manera que podrá almacenar una tarea en cualquier hora del día.
- ★ **Columnas:** Contendrán los días correspondientes al mes en donde se encuentren por lo cual pueden presentar una variación de días dependiendo del mes en el que se encuentre.

- ★ **Celdas:** Este apartado es la conexión que se da de un día a una hora, dentro de este nodo de la matriz se maneja un apuntador hacia una lista simplemente enlazada, esta lista simplemente enlazada almacenará todas las tareas que coinciden con la fecha y hora.

De esta manera se almacenan los recordatorios que un estudiante desee almacenar, algo importante a tener en cuenta es que no es necesario que dentro de la matriz dispersa se encuentre todos los días o horas especificados, sólo deberán aparecer los días y horas que contengan actividades por realizar.

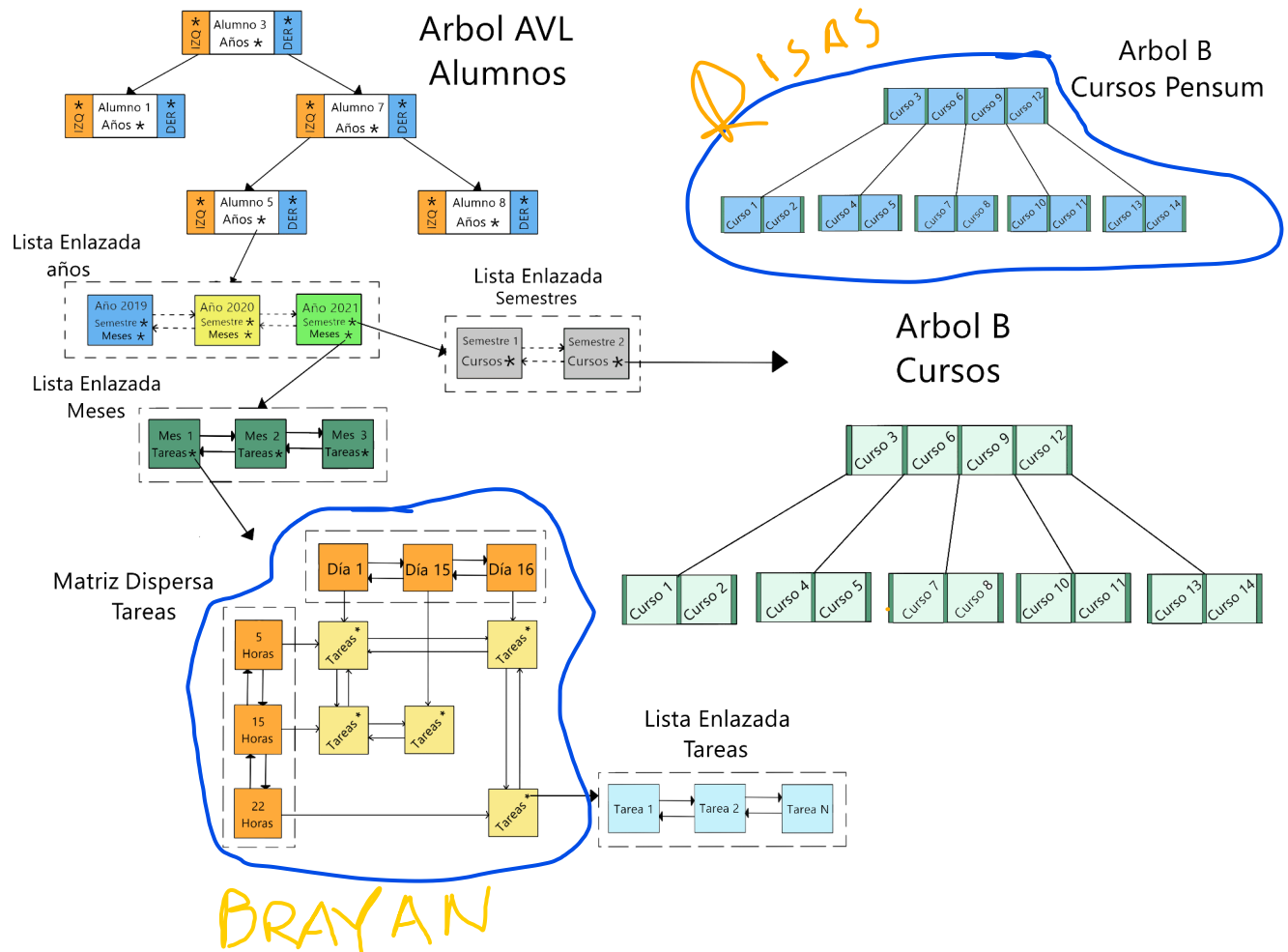
La carga masiva de este apartado se realiza al mismo tiempo que la carga masiva de los estudiantes, esto porque dependen del mismo archivo para su carga de datos.

Semestres

Luego de ya tener almacenados los años que un estudiante desea guardar, se debe proceder a almacenar los semestres, para ello se debe utilizar una lista simplemente enlazada que permita almacenar los dos semestres que hay en un año, algo importante que se debe establecer es que esta lista no debe permitir el almacenamiento de datos repetidos y debe mantener un tamaño no mayor a 2 posiciones, esto permite restringir el almacenamiento de solo dos semestres, además, que se debe tener en cuenta que esta estructura dentro de cada nodo debe de contener un apuntador a un árbol B que almacenará los cursos que un estudiante recibió en el semestre, por ello al construir la estructura se debe tener en cuenta este aspecto.

Arquitectura General de Estructuras

A continuación se muestra la arquitectura general de las estructuras que se manejan en la fase actual.



Estructuras

- **Arbol AVL**
 - Utilizado para almacenar alumnos
- **Listas Enlazadas**
 - Utilizado para almacenar Años
 - Utilizado para almacenar Meses
 - Utilizado para almacenar Semestres
 - Utilizado para almacenar Tareas
- **Matriz Dispersa**
 - Utilizado para llevar el control de Tareas
- **Arbo B**
 - Utilizado para Registrar Cursos del Pensum
 - Utilizado para Registrar Cursos del Estudiante.

Árbol B de Cursos

Por la cantidad de cursos que se manejan a nivel de pensum de la carrera de Ingeniería , se utilizará la estructura de árbol B de **orden 5**, donde cada nodo de una página representa 1 curso y deberá guarda la siguiente información:

Datos del Curso:

- Código del curso
- Nombre
- Número de Créditos
- Códigos de cursos como prerrequisito
- Si el curso es Obligatorio o no lo es

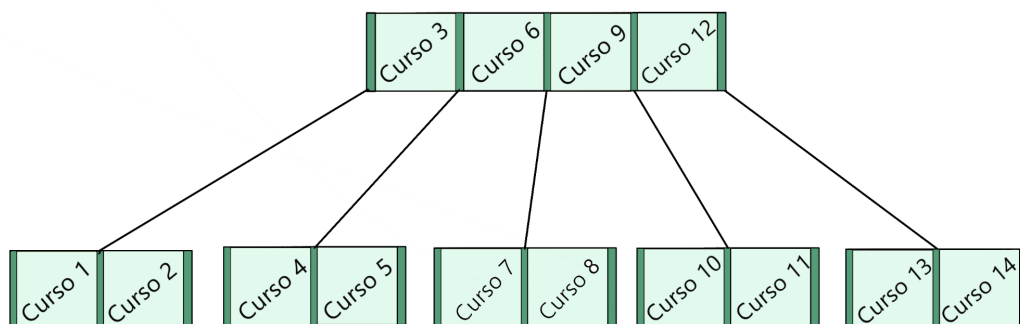
Cada Escuela de la Carrera de Ingeniería maneja su pensum de una manera distinta, de modo que en esta ocasión se manejan cursos de la escuela de ciencias y sistemas.

Nota

Tener en cuenta que, para utilizar la estructura de Árbol B se debe utilizar una estructura para alojar los cursos del Pensum y se debe utilizar otra estructura de árbol B para alojar los cursos de un estudiante, los cuales ha decidido llevar.

Cursos del Pensum.

A continuación se muestra la estructura de Árbol B para alojar cursos correspondientes a un pensum de Ingeniería.

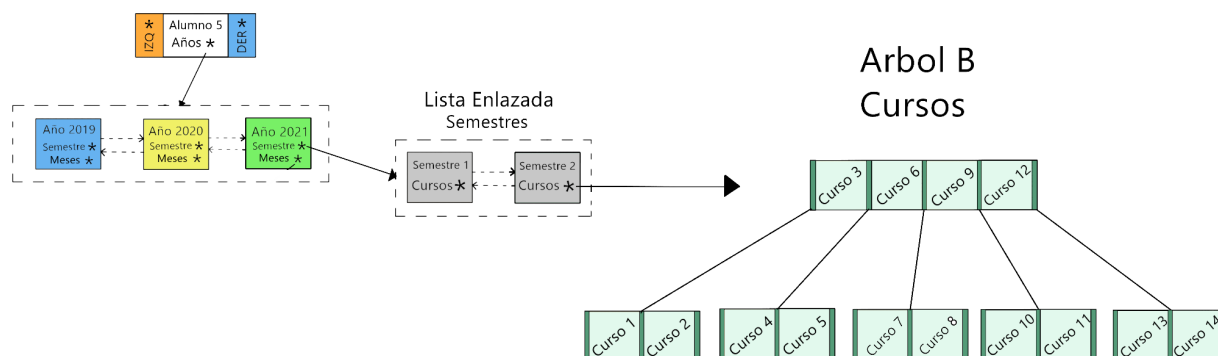


La estructura para la carga masiva se presenta a continuación.

[Cursos Pensum.json](#)

Cursos del alumno.

A continuación se muestra la manera de utilizar un Árbol B para almacenar cursos en donde el estudiante previamente tuvo que realizar la acción de escoger los cursos que desea llevar.



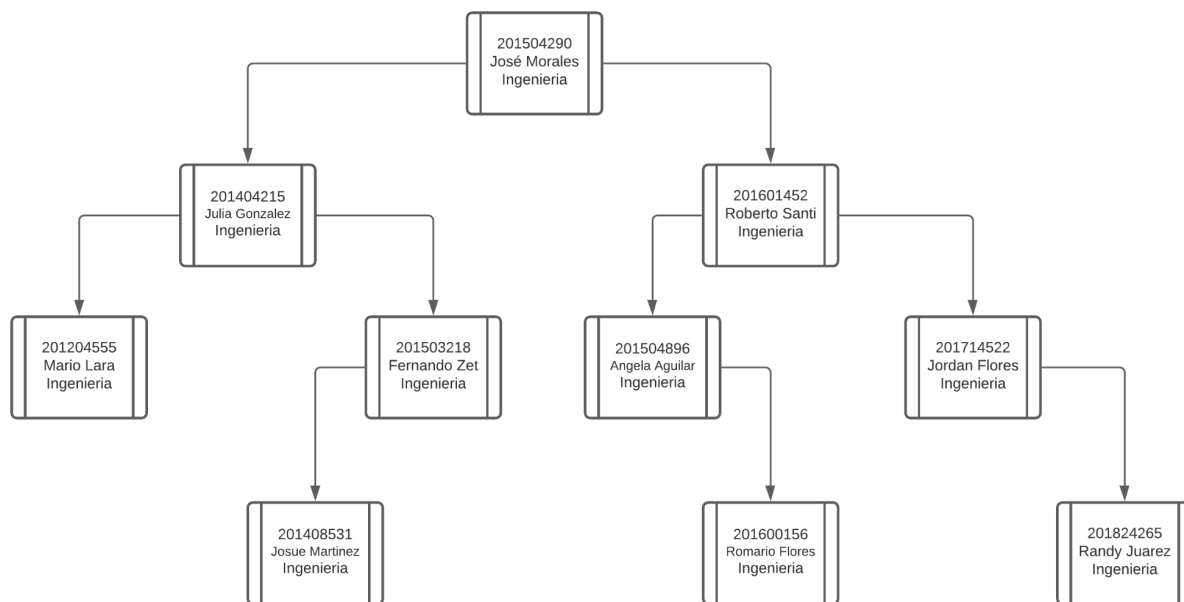
La estructura para la carga masiva se presenta a continuación.

[Cursos Estudiante.json](#)

Reportes

Árbol AVL de Estudiantes

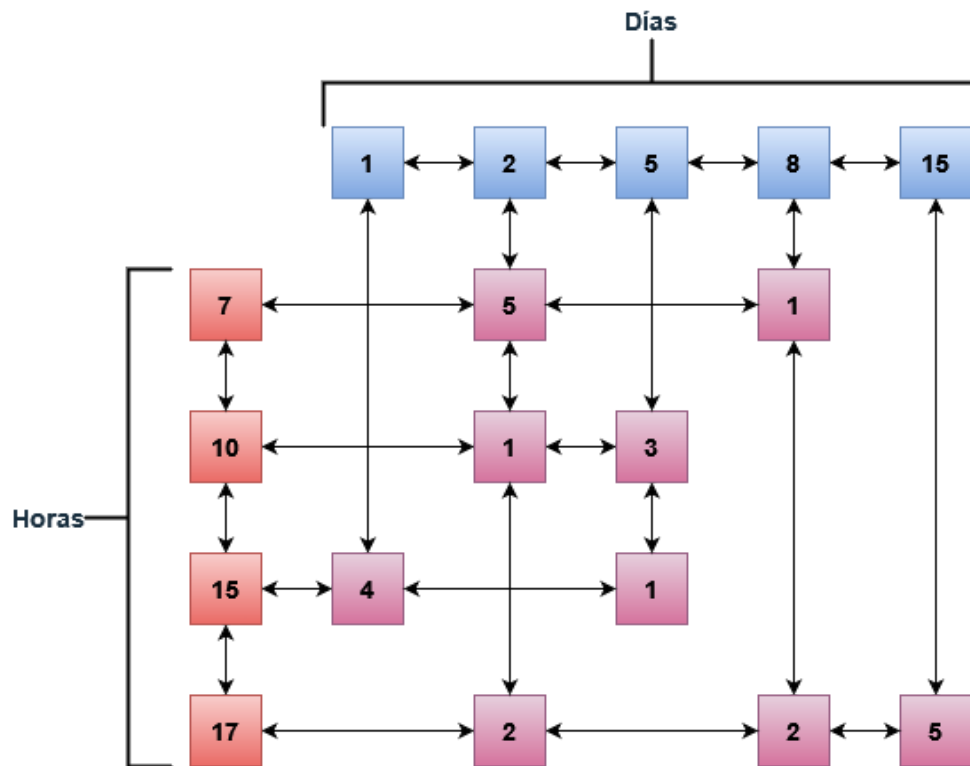
El reporte del árbol AVL de estudiantes consiste en la visualización de todos los estudiantes que han sido ingresados a la aplicación, en el cual se visualizará el carnet, nombre y carrera del estudiante.



Matriz Dispersa de Tareas

El reporte consiste en la generación de un gráfico de una matriz Dispersa, para ello el estudiante debe enviar una solicitud al servidor.

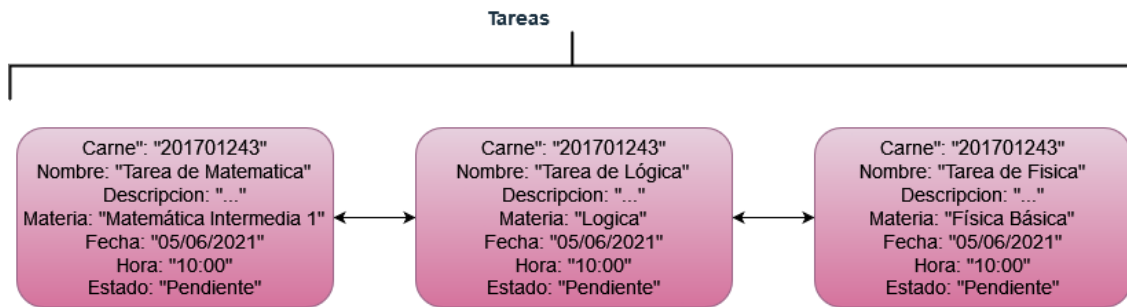
En el momento que el servidor reciba dichos parámetros debe proceder a buscar el estudiante con el carné seleccionado, luego buscar el año correspondiente y por último ubicar el mes, esto para poder obtener la matriz dispersa correspondiente a los datos que se recibieron, como respuesta debe generar un gráfico de la matriz, en donde cada celda correspondiente solo debe mostrar la cantidad de tareas que se tienen dentro de la lista de tareas. En el caso que no existiese una matriz que cumpla con los parámetros de la solicitud, se debe informar al estudiante para que proceda a cambiar la solicitud, si existiese debe mostrar la matriz. A continuación se muestra un ejemplo de cómo podría estar construida la matriz.



Lista de tareas

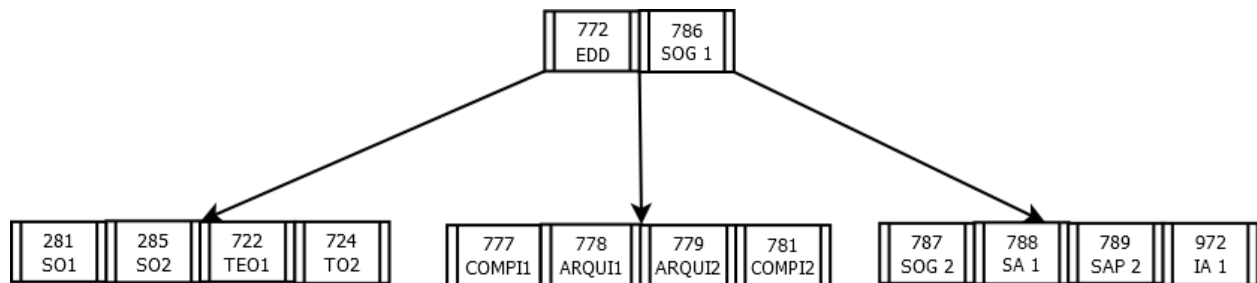
El reporte consiste en la generación de una lista que contenga las tareas correspondientes a una hora y fecha específica, para ello el estudiante que desee obtenerlo debe enviar una petición al servidor.

Algo importante que se debe mencionar es que si no existiesen tareas en ese día se le debe informar al estudiante, para que cambie la consulta a realizar, en el caso de que si existan tareas se debe devolver una lista con las tareas existentes, tal y como se muestra a continuación.



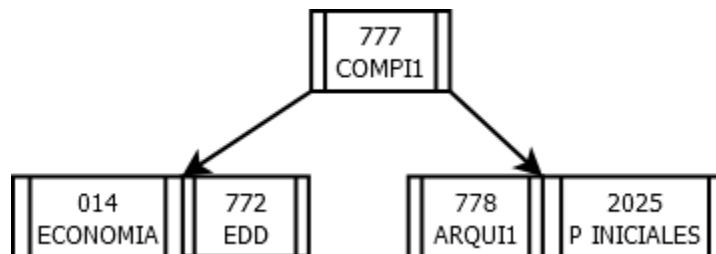
Árbol B de Cursos(General)

Este reporte consiste en la generación de un gráfico de un árbol B, el cual muestra el nombre y código del curso en el árbol donde se almacenaron todos los cursos del pensum.



Árbol B de Cursos de un estudiante en un semestre específico

Este reporte recibe como parámetro el carnet de un estudiante, un año y un semestre, con el cual se buscará el árbol B de cursos que tenía asignados ese semestre. Se deberá graficar el árbol B.



Peticiones al servidor

Carga masiva

Para realizar la carga masiva de las diferentes estructuras se debe seguir el mismo formato, el cual se especifica a continuación:

El endpoint requerido para realizar la carga masiva será solamente uno de tipo **POST**, siendo el siguiente: **localhost:puerto/carga**. Esta petición contendrá los siguientes parámetros dentro de ella:

```
{
  "tipo": "estudiante|recordatorio|curso",
  "path": "direccion_de_archivo_de_carga"
}
```

Como se puede visualizar tendrá dos parámetros, el primero es el tipo en el cual se indica que tipo de carga masiva se realizará, y como segundo parámetro tenemos el path, el cual nos muestra la dirección donde se encuentra el archivo de entrada.

Reportes

Este apartado contiene todos los reportes mencionados con anterioridad permitiendo obtener cada reporte en un único endpoint (tipo **GET**), el cual se muestra a continuación: **localhost:puerto/reporte**. Esta petición contendrá una serie de parámetros que variarán dependiendo del tipo de reporte que se genere, a continuación se muestran todos los formatos que puede tener este endpoint.

AVL de estudiantes

El primer reporte que se puede generar es el AVL de estudiantes, y los parámetros para poder generarlo son los siguientes:

```
{
  "tipo": 0
}
```

Matriz de Tareas

El siguiente reporte que se puede tener es la matriz de tareas, conteniendo los siguientes parámetros para su generación.

```
{  
  "tipo": 1,  
  "carnet": "201701243",  
  "año": "2019",  
  "mes": 5  
}
```

Lista de tareas

El tercer reporte que se puede generar es la lista de tareas, el cual necesita los siguientes parámetros para su correcto funcionamiento:

```
{  
  "tipo": 2,  
  "carnet": "201701243",  
  "año": "2019",  
  "mes": 6,  
  "dia": 5,  
  "hora": 10  
}
```

Árbol B de Cursos(General)

El cuarto reporte a generar es el árbol B de cursos, el cual contiene todos los cursos de la carrera, utilizando para ello los siguiente parámetros.

```
{  
  
  "tipo": 3  
  
}
```

Árbol B de Cursos de un estudiante en un semestre específico

El quinto reporte que se puede generar es un árbol B de cursos por estudiante y semestre específico, utilizando para obtenerlo los siguientes parámetros.

```
{  
  
  "tipo": 4,  
  
  "carnet": "201701243",  
  
  "año": "2019",  
  
  "semestre": "1"  
  
}
```

CRUD Estudiantes

Este apartado nos indicará todas las operaciones que se pueden realizar con la estructura de tipo estudiante. Este apartado sólo contendrá un endpoint en el cual se podrá variar el tipo de petición para obtener una operación distinta.

El endpoint a utilizar es: **localhost:puerto/estudiante**, y las operaciones que se pueden realizar son las siguientes.

Crear un estudiante

Para poder crear un estudiante se deben trasladar todos los parámetros correspondientes a la estructura que presenta, los cuales se mostrarán a continuación, además el tipo de petición que se debe realizar es **POST**.


```
{
  "carnet": "201701243",
  "DPI": "3145517870101",
  "nombre": "Alvin Hockett",
  "carrera": "Ingeniería en Ciencias y Sistemas",
  "correo": "AlvinHockett_60@outlook.org",
  "password": "DGHS2Xi",
  "creditos": 152,
  "edad": 25
}
```

Modificar estudiante

Para poder modificar un estudiante se deben trasladar todos los parámetros correspondientes a la estructura que presenta, los cuales se mostrarán a continuación, además el tipo de petición que se debe realizar es **UPDATE**.

```
{
  "carnet": "201701243",
  "DPI": "3145517870101",
  "nombre": "Alvin Hockett",
  "carrera": "Ingeniería en Ciencias y Sistemas",
  "correo": "AlvinHockett_60@outlook.org",
  "password": "DGHS2Xi",
  "creditos": 152,
  "edad": 25
}
```

Eliminar un estudiante

Para poder eliminar un estudiante se debe trasladar el número de carné que presenta, el cual se mostrará a continuación, además el tipo de petición que se debe realizar es **DELETE**.

```
{  
  
  "carnet": "201701243"  
  
}
```

Obtener información de un estudiante

Para poder obtener la información de un estudiante se debe trasladar el número de carné que presenta, el cual se mostrará a continuación, además el tipo de petición que se debe realizar es **GET**.

```
{  
  
  "carnet": "201701243"  
  
}
```

CRUD Recordatorios

Este apartado nos indicará todas las operaciones que se pueden realizar con la estructura de tipo Matriz dispersa. Este apartado sólo contendrá un endpoint en el cual se podrá variar el tipo de petición para obtener una operación distinta.

El endpoint a utilizar es: **localhost:puerto/recordatorios**, y las operaciones que se pueden realizar son las siguientes.

Crear Recordatorio

Para poder crear un recordatorio se deben trasladar todos los parámetros correspondientes a la estructura que presenta, los cuales se mostrarán a continuación, además el tipo de petición que se debe realizar es **POST**.

```
{
  "Carnet" : "201914113",
  "Nombre" : "Tarea de Matematicas",
  "Descripcion" : "Realizar la tarea previa la parcial",
  "Materia": "Matematica intermedia 1",
  "Fecha": "01/01/2021",
  "Hora": "8:00",
  "Estado": "incumplido"
}
```

Modificar Recordatorio

Para poder modificar un recordatorio se deben trasladar todos los parámetros correspondientes a la estructura que presenta, los cuales se mostrarán a continuación, además el tipo de petición que se debe realizar es **UPDATE**.

```
{
  "Carnet" : "201914113",
  "Nombre" : "Tarea de Matematicas",
  "Descripcion" : "Realizar la tarea previa la parcial",
  "Materia": "Matematica intermedia 1",
  "Fecha": "01/01/2021",
  "Hora": "8:00",
  "Estado": "incumplido"
}
```

Obtener información de un Recordatorio

Para poder obtener la información de un recordatorio se debe trasladar la hora y la fecha, el cual se mostrará a continuación, además el tipo de petición que se debe realizar es **GET**.

```
{
  "Fecha" : "01/08/2021",
  "Hora" : "8:00"
}
```

Eliminar Recordatorio

Para poder eliminar un recordatorio se debe trasladar la fecha y la hora, el cual se mostrará a continuación, además el tipo de petición que se debe realizar es **DELETE**.

```
{
  "Fecha" : "01/08/2021",
  "Hora" : "8:00"
}
```

CRUD Cursos

Este apartado nos indicará todas las operaciones que se pueden realizar con la estructura de tipo Árbol B. Este apartado sólo contendrá un endpoint en el cual se podrá variar el tipo de petición para obtener una operación distinta.

El endpoint para los cursos del estudiante a utilizar es: **localhost:puerto/cursosEstudiante**, y el endpoint para los cursos del pensum a utilizar es: **localhost:puerto/cursosPensum** operación de crear curso es la siguiente:

Crear Curso

Para poder crear un recordatorio se deben trasladar todos los parámetros correspondientes a la estructura que presenta, los cuales se mostrarán a continuación, además el tipo de petición que se debe realizar es **POST**.

Cursos de Estudiantes

```
{
  "Estudiantes": [
    {
      "Carnet": "201901423",
      "Años": [
        {
          "Año": "2020",

```

```
"Semestres": [  
  {  
    "Semestre": "2",  
    "Cursos": [  
      {  
        "Codigo": "774",  
        "Nombre": "Sistemas de Bases de Datos  
1",  
        "Creditos": 5,  
        "Prerequisitos": "773",  
        "Obligatorio": true  
      },  
      {  
        "Codigo": "281",  
        "Nombre": "Sistemas Operativos 1",  
        "Creditos": 4,  
        "Prerequisitos": "781,778",  
        "Obligatorio": true  
      },  
      {  
        "Codigo": "2036",  
        "Nombre": "Práctica Intermedia",  
        "Creditos": 0,
```



```
{
  "Codigo": "039",
  "Nombre": "Deportes 1",
  "Creditos": 1,
  "Prerequisitos": "",
  "Obligatorio": false
},
{
  "Codigo": "348",
  "Nombre": "Quimica General 1",
  "Creditos": 3,
  "Prerequisitos": "",
  "Obligatorio": true
},
{
  "Codigo": "103",
  "Nombre": "Matemática Básica 2",
  "Creditos": 7,
  "Prerequisitos": "101",
  "Obligatorio": true
},
{
  "Codigo": "005",
```

```
    "Nombre": "Técnicas de Estudio e Investigación",
    "Creditos": 3,
    "Prerequisitos": "",
    "Obligatorio": true
},
{
    "Codigo": "960",
    "Nombre": "Matemática Cómputo 1",
    "Creditos": 5,
    "Prerequisitos": "103",
    "Obligatorio": true
},
{
    "Codigo": "107",
    "Nombre": "Matemática Intermedia 1",
    "Creditos": 10,
    "Prerequisitos": "103",
    "Obligatorio": true
}
]
}
```


Consideraciones

1. El lenguaje a utilizar será Python.
2. Puerto por defecto **3000**.
3. La carpeta de la generación de los reportes debe ser creada en el escritorio, y debe llamarse **Reportes_F2**.
4. Las estructuras serán realizadas por el estudiante.
5. IDE a utilizar es libre (Recomendaciones: **Visual Studio Code, PyCharm, Sublime Text**).
6. La entrega será por medio de la plataforma UEDI.
7. El estudiante debe tener un repositorio privado en github con el nombre **EDD_SmartClass_#Carné** y agregar a su tutor como colaborador al repositorio del proyecto (Cada tutor les hará llegar su estudiante).
8. Será calificado del último commit realizado antes de la fecha y hora de entrega.
9. Las copias totales o parciales **serán penalizadas con nota de 0 puntos y reportadas** ante la escuela de ciencias y sistemas.
10. **Se deberán graficar todas las estructuras de la fase con graphviz.**
11. **Fecha de entrega: 26 de septiembre de 2021 antes de las 23:59 horas.**