

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN DE CIENCIAS DE LA INGENIERÍA



Proyecto #1
Creación de modelo y simulación de
Generación de horarios

Modelación y simulación

PRESENTADO POR:

Umaña de León, William Rodrigo

201931448

DOCENTE:

Ing. Pedro Domingo

QUETZALTENANGO – QUETZALTENANGO – GUATEMALA

18-09-2023

Índice

Índice	2
Descripción del Proyecto	4
Requerimientos	5
Requerimientos funcionales	5
Requerimientos no funcionales	5
Requerimientos extras	5
Formulación del modelo	6
Objetivo	6
Identificación de variables	6
Aleatorización	8
Función del sistema	8
Restricciones	9
Diagrama cualitativo	10
Comparación de modelos	10
Implementación del modelo	13
Librerías de python Utilizadas	13
Utilización de mysql	13
Configuración de Base de datos	14
Creación de modelo	14
Repositorio	15
Utilización de bottle	15
Utilización de conexión a Mysql	15
Guardado de variables	17
Control de datos	17
Archivo de subida de datos	17
Modelo implementado	18
Creación de distribucion de períodos	18
Analizador de horario	20
Utilizacion del profesor	24
Advertencias	26
Verificación del modelo	26
Resultado verificacion 1	28
Resultado verificación 2	29
Resultado verificación 3	30
Resultado verificación 4	32
Experimentación	34
Experimento #1	34
Resultados obtenidos	36

Interpretación	36
Experimento #2	37
Resultados Obtenidos	37
Interpretación de los resultados	38
Experimento #3	39
Interpretación de los resultados	44
Manual de usuario	45
Referencias	51

Descripción del Proyecto

El siguiente enunciado es el proporcionado por el ingeniero a cargo de la supervisión y calificación del proyecto:

Se le contrató como ingeniero en sistemas en la organización “la usurpación”, la cual se dedica a la educación superior, donde necesitan una solución informática que les ayude a tomar las decisiones sobre la creación de horarios de clases, actualmente invierten mucho tiempo en dicha actividad por lo que quisieran tener diferentes propuestas generadas automáticamente y elegir la que mejor se les ajuste a sus necesidades, entre las condiciones de como funciona la organización están las siguientes:

- Disponen de un espacio finito de salones (s).
- Cada salón tiene una disponibilidad finita de escritorios para los estudiantes (e).
- Existen diferentes carreras (c) y cada carrera tiene sus propias materias (m).
- Tienen profesores encargados de impartir las materias (p), estos pueden estar asignados a materias fijas o pueden cubrir otras materias si poseen las cualificaciones y hace falta personal.
- Tienen asignación de cursos desde el inicio (a), por lo que se sabe que materias cursarán los estudiantes en el semestre.

Características generales:

- Este deberá ser alimentado por archivos de carga de base de datos (scripts) indicando las diferentes variables del sistema (s, e, c, m, p, a) y otras que considere necesarias.
- El sistema tendrá las opciones de generar un horario tomando diferentes criterios de prioridad, por ejemplo: en base a las materias disponibles, en base a la disponibilidad de espacios en los salones, en base al horario de contratación de los profesores, entre otros.
- El sistema podrá indicar advertencias en los horarios asignados.

Requerimientos

Requerimientos funcionales

- El sistema debe tener la capacidad de generar automáticamente horarios de clases en base a diferentes criterios de prioridad, como la disponibilidad de salones, la disponibilidad de profesores, la disponibilidad de escritorios para estudiantes
- El sistema debe ser capaz de asignar profesores a las materias de acuerdo con sus cualificaciones y la necesidad de personal, ya sea asignándoles materias fijas o materias adicionales.
- El sistema debe proporcionar la opción de generar múltiples propuestas de horarios, según un índice de eficiencia para que la persona a cargo pueda tomar el que mejor le convenga
- El sistema debe ser capaz de mostrar advertencias o conflictos en los horarios asignados

Requerimientos no funcionales

- El sistema debe ser eficiente y rápido en la generación de horarios
- El sistema debe ser fácil de usar y contar con una interfaz intuitiva para que los usuarios puedan interactuar con él de manera efectiva.
- El sistema debe ser escalable para poder gestionar grandes cantidades de datos
- El sistema debe tener la capacidad de realizar copias de seguridad de los horarios generados

Requerimientos extras

- El sistema debe aceptar archivos de carga, que contengan información sobre salones, escritorios, carreras, materias, profesores y asignaciones de cursos.
- El sistema debe generar horarios de clases en un formato legible para que la organización pueda revisarlos y tomar decisiones.
- El sistema debe mostrar advertencias y conflictos de manera clara

Formulación del modelo

Objetivo

El objetivo principal es desarrollar un sistema que, a partir de la información proporcionada y siguiendo criterios de prioridad, genere horarios de clases eficientes y sin conflictos. El sistema debe ser capaz de proponer diferentes alternativas de horarios para que la organización pueda elegir la que mejor se adapte a sus necesidades.

Además, el sistema debe ser alimentado por archivos de carga de base de datos que contienen información sobre las variables del sistema (salones, escritorios, carreras, materias, profesores, asignaciones) y otras que se consideren necesarias. Asimismo, el sistema debe tener la capacidad de señalar advertencias en caso de que se produzcan conflictos en los horarios asignados.

El propósito final de este proyecto es mejorar la eficiencia en la planificación de horarios, reducir la carga de trabajo manual y garantizar una asignación de recursos óptima para brindar una experiencia de aprendizaje efectiva para los estudiantes y un entorno de trabajo adecuado para los profesores y personal de la organización.

Identificación de variables

- Salones (s): Representa los diferentes espacios físicos disponibles en la organización para impartir clases. Cada salón tiene una capacidad limitada.
- Escritorios para estudiantes (e): Indica la disponibilidad de escritorios o asientos para los estudiantes en cada salón. La cantidad de escritorios puede variar de un salón a otro.
- Carreras (c): Representa las diferentes disciplinas académicas o programas de estudio ofrecidos por la organización. Pueden haber múltiples carreras.
- Materias (m): Son las asignaturas o cursos individuales que forman parte de cada carrera. Cada carrera tiene sus propias materias, y estas pueden tener requisitos de horario específicos.

- Profesores (p): Representa al personal docente encargado de impartir las materias. Cada profesor puede estar asignado a materias específicas y puede tener su propia disponibilidad de horario.
- Asignaciones de cursos (a): Indica las materias que los estudiantes deben cursar en un semestre o período académico determinado. Esta asignación se conoce desde el inicio y debe ser tomada en cuenta al crear los horarios.
- Horarios de clases: Estas son las variables que se deben determinar mediante el sistema. Los horarios serán representados por matrices que contendrán periodos donde estarán asignados los cursos, profesores y alumnos.
- Periodo: El periodo es la forma en que estará compuesto el horario, en él se encontrará una hora a la que se da, un salón, el curso que se impartirá, el profesor, y los alumnos asignados.
- Hora de inicio y finalización del horario: Es la hora a la que empezará el horario a tomar en cuenta los periodos, este respondería a una aproximación según la duración del periodo
- Duración del periodo: Esta es la cantidad de minutos que tardará un periodo, según esta cantidad se generarán los periodos dentro del horario de inicio y finalización que se dio.
- Semestre del curso: Aunque esta variable sea intrínseca del curso o materia se tiene que agregar aquí ya que hay que hacer la aclaración de que existen varios horarios donde solo se asignan los cursos de semestre impar o par.

Estas son las variables más importantes de todo el sistema, y son las que el usuario tiene un control casi total de ellas, ya que serán sobre las que se generarán los horarios.

Ahora bien existen otro tipo de variables sobre las que el usuario no tiene un total control pero de las que sí puede escoger.

- Tipo de ordenamiento: El tipo de ordenamiento corresponde a la manera en que se ordenarán los cursos para asignarse, ya que estos se tienen pensado que se asignan en un modo de fila, el primero en llegar el primero en asignarse.

De los tipos de ordenamiento se tienen las siguientes consideraciones:

- Asignación de curso de forma ascendente según su código de asignación.
- Asignación de curso de forma descendente según su código de asignación.
- Asignación de curso según la cantidad de estudiantes que tiene asignados
- Asignación de curso según una prioridad asignada por el usuario

De estos dos últimos se toma en cuenta la aleatorización de la asignación por si en caso hayan cursos con igual número de estudiantes o prioridad.

- Cantidad de periodos a crear: Esta es una opción que aparece únicamente cuando se esté trabajando con los tipos de ordenamiento que conlleven aleatorización ya que esa forma podrán ver varios horarios de una sola forma de ordenamiento.

Aleatorización

Consiste en que tanto la asignación del material experimental como el orden en que se realizan las pruebas individuales o ensayos se determinan aleatoriamente y sirve para:

- Garantizar la validez de la estimación del error experimental.
- Garantizar la independencia de los errores o que las observaciones sean variables aleatorias independientes. Esto es necesario para obtener pruebas de significancia válidas y estimados de intervalos.
- Eliminar el sesgo de tal manera que no se desfavorece o discrimine a los tratamientos y permite cancelar los efectos de factores extraños que pudieran estar presentes.

Función del sistema

La función objetivo en este problema de programación de horarios tiene como objetivo principal optimizar ciertos aspectos del proceso de asignación de horarios. Algunos de estos objetivos son:

- Minimizar el costo de asignación de recursos
- Maximizar la satisfacción de estudiantes y profesores

- Minimizar las superposiciones de horarios
- Maximizar la utilización de recursos
- Maximizar la calidad de los horarios

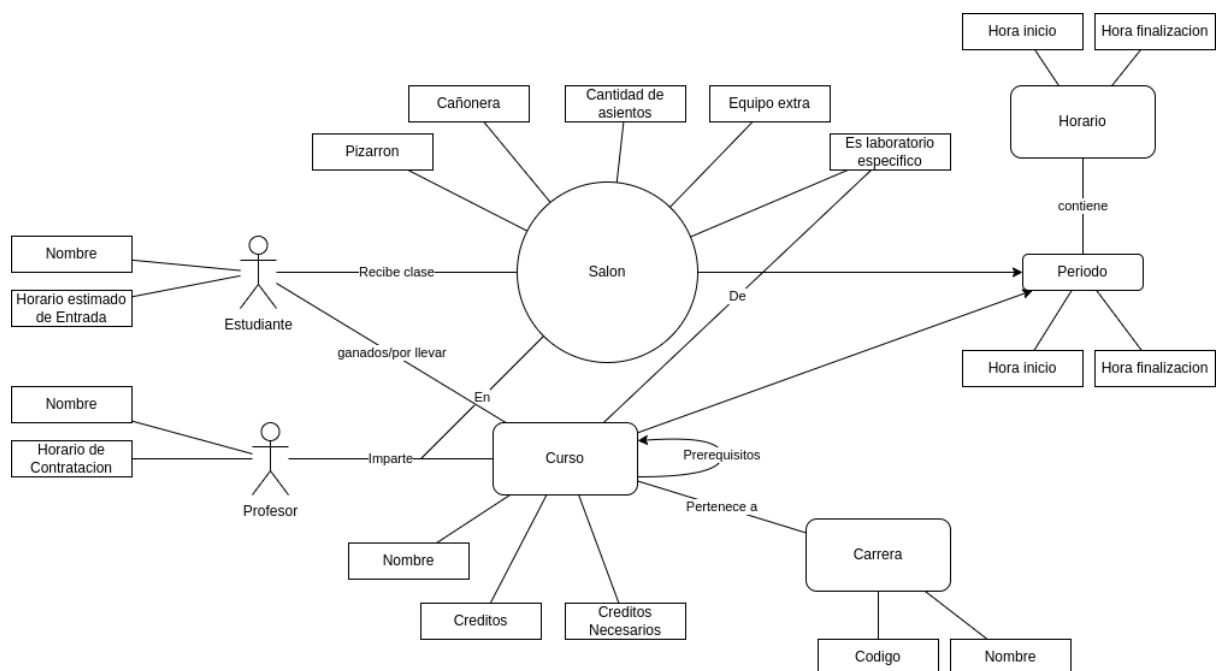
Aunque todos estos objetivos por separado conforman una aproximación al modelo completamente diferente se tratan de abordar una parte de cada uno para poder brindar una mejor calidad en cuanto a la asignación de los horarios.

Restricciones

- Restricción de salones: Cada salón tiene una capacidad máxima y debe estar disponible en ciertos momentos.
- Restricción de duración de las materias: Cada materia tiene una duración específica que debe ser considerada en la programación de horarios.
- Restricción de disponibilidad de profesores: Los profesores tienen horarios disponibles en los que pueden enseñar, y estos horarios deben ser respetados.
- Restricción de cualificaciones: Los profesores solo pueden enseñar materias para las que estén cualificados. Aunque se piensa implementar una forma para que el profesor sea asignado a un curso de su carrera.
- Restricción de superposición de horarios: No deben existir clases superpuestas si son del mismo semestre.
- Restricción de superposición de horarios de profesores: No deben existir clases superpuestas si el profesor está dando clases en otro salón al mismo tiempo, lo cual sería imposible.
- Restricción de horarios de trabajo de profesores: Los profesores pueden tener restricciones de tiempo en función de sus horarios de trabajo. No se toman en cuenta que ellos tengan alguna hora en específico que no puedan dar.

Diagrama cualitativo

Este diagrama cualitativo trata de ejemplificar la forma en que funciona el sistema sin necesidad de mostrar algún cálculo o función que se ejecutara dentro de este para asignar los cursos más que sus relaciones entre las variables del sistema.



Comparación de modelos

Como bien se sabe se está realizando este modelo con el objetivo de tomar en cuenta lo que realiza la universidad y poder realizar una comparación sobre lo que hacen actualmente y lo que podría hacer un modelo de simulación. De tal forma que se trata de explicar con un diagrama de procesos lo que se realiza actualmente la universidad y lo que se pretende realizar con el modelo.

Diagrama de creación de horario actual

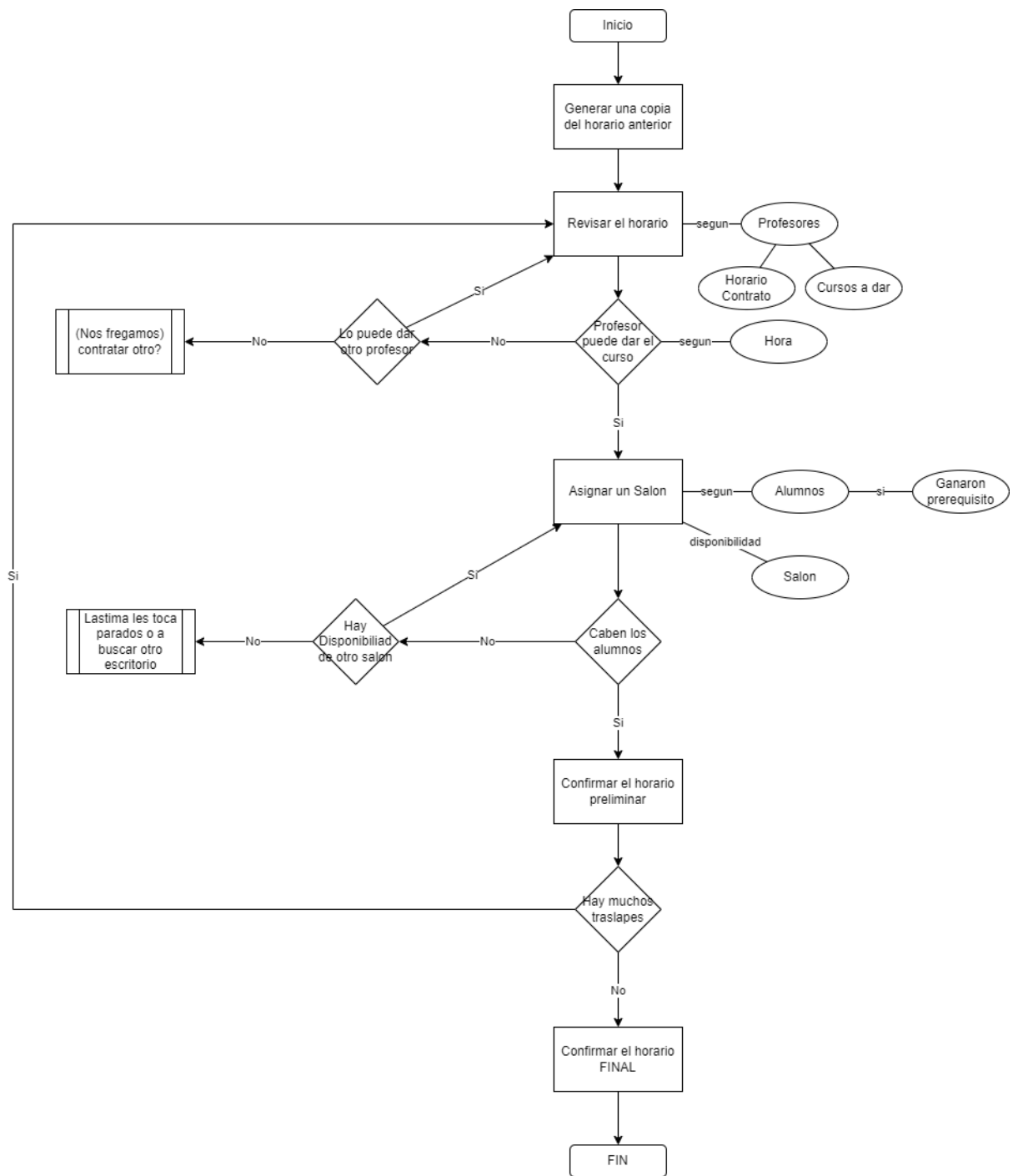
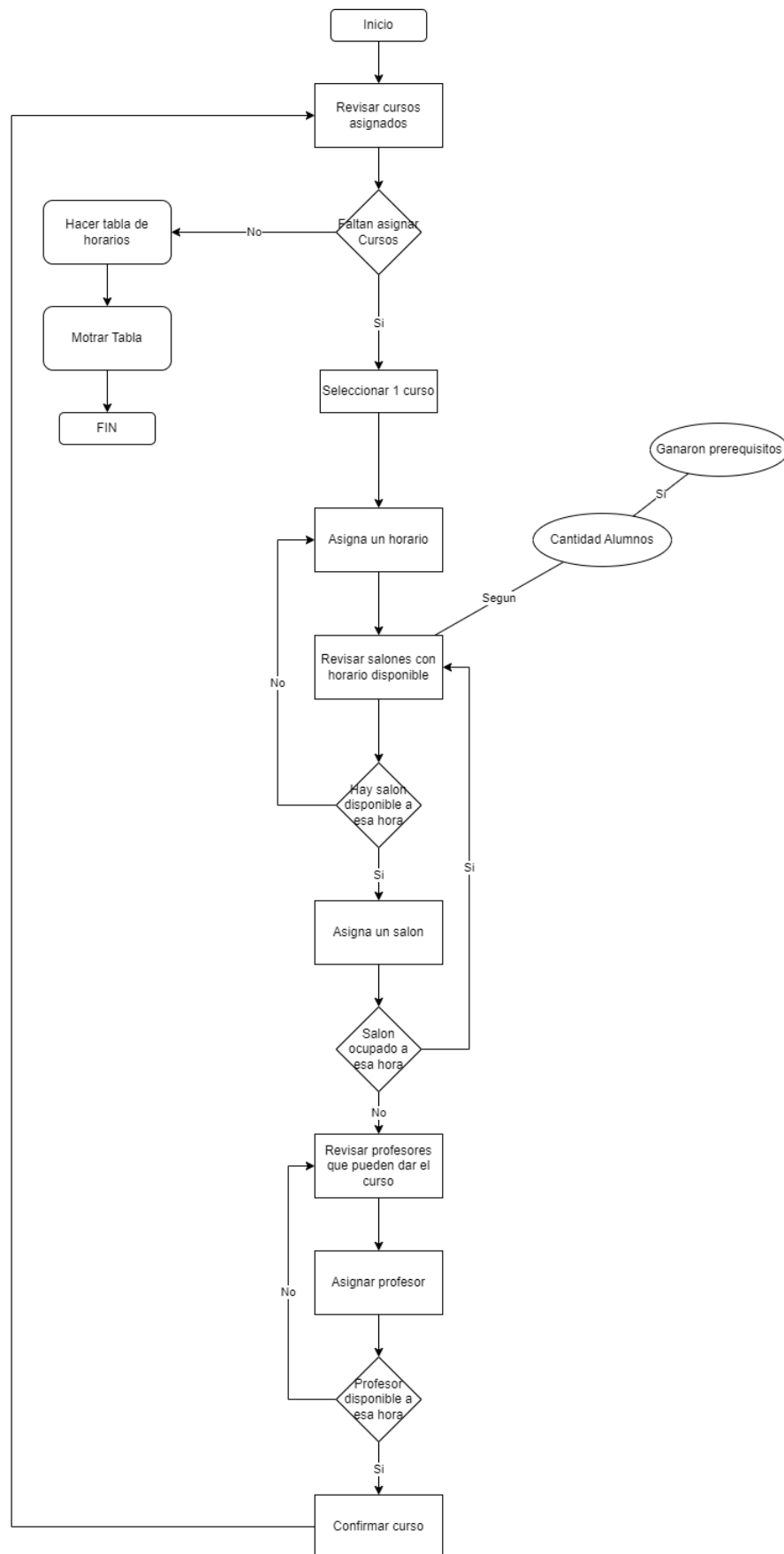


Diagrama de creación de horario mediante el sistema



Implementación del modelo

Al tener ya una imagen sobre lo que se tiene que trabajar, así como también un análisis sobre lo que hay que presentar y manejar, se proyecta utilizar las herramientas:

- Python: un ambiente de trabajo trabajado y manejado puramente con python
- Mysql: Para manejar un histórico sobre los horarios que se han creado
- Navegador Web: Para poder visualizar los horarios que se crean, así como también poder presentar al usuario el formulario para la elección del tipo de asignación, la subida de los datos, etc.

Conforme a como se trabajó el proyecto se describen las librerías y herramientas que se utilizaron:

Librerías de python Utilizadas

- Datetime
- Copy
- Random
- matplotlib.pyplot : Solo se utilizó al inicio para generar una imagen de los horarios
- bottle: Este es un micro web-framework, que permite la creación de un mini motor para poder visualizar html,css y js en un navegador web, así como el manejo de las rutas. En pocas palabras es un mini-flask.
- jsonpickle: Esta librería sirve para poder hacer una o varias clases en formato json y así poder guardarlas en la base de datos.
- mysql.connector

Utilización de mysql

Cabe mencionar que este modelo se realizó una vez ya generados los horarios y realizar una base de datos en concreto para cada uno de los datos que el usuario estará ingresando parecía una pérdida de espacio además que estos datos serán volátiles por lo que se tomó la decisión de guardar únicamente el resultado final (los horarios), de tal forma que la base de datos se compone de la siguiente manera:

Configuración de Base de datos

Estos son los comandos utilizados para la creación de la base de datos en mysql, así como también el usuario que se utilizara para conectar python con esta base de datos.

```
-- Crear la base de datos
CREATE DATABASE IF NOT EXISTS base_Horarios;

-- Seleccionar la base de datos
USE base_Horarios;

-- Crear la tabla horario
CREATE TABLE IF NOT EXISTS horario (
    id_horario INT AUTO_INCREMENT PRIMARY KEY,
    fecha_hora DATETIME,
    nombre VARCHAR(255),
    datos JSON
);

-- Creación de usuario
CREATE USER 'adminHorarios'@'localhost' IDENTIFIED BY
'password123';
-- Agregar privilegios al usuario
GRANT ALL PRIVILEGES ON base_Horarios.* TO
'adminHorarios'@'localhost';
FLUSH PRIVILEGES;
```

Cabe mencionar que se utilizó mariaDB como motor para utilizar mysql ya que la distribución de linux que se utilizó tiene como uso predeterminado la utilización de mariaDB y no de mysql en general. Por lo que si ocurre algún error con la base de datos se recomienda utilizar mariaDB.

Creación de modelo

Ahora que ya se tiene la descripción de todo lo que se tiene que realizar y una aproximación sobre lo que hay que basarse para realizar el modelo se crea el modelo utilizando python como motor básico de este proyecto, cabe destacar que se utilizó este lenguaje como orientado a objetos, por lo que está distribuido por clases y creación de objetos.

Repositorio

Se utilizó github para guardar el código y configuración del sistema:

<https://github.com/waliray123/Proyecto1Modela2>

Utilización de bottle

bottle es un microframework web, lo que significa que es extremadamente ligero y simple de usar. Esto lo hace ideal para proyectos pequeños y como este es un proyecto que no necesita una gran interfaz tamaño empresarial o algo por el estilo se decidió utilizar este modelo.

La clase que utiliza este modelo es la clase principal “initHost” quien es la que prende por decirlo de alguna manera, todo el proyecto y lo pone a andar. Un poco de su configuración es la siguiente:

```
from bottle import route, get, run, template, static_file,
redirect, request
esNuevo = 0
esPrimero = 1
@route('/')
def index():
    global esPrimero, esNuevo
    esPrimero = 1
    esNuevo = 0
    return template('templates/index.tpl')

run(host='localhost', port=8080, debug=True)
```

Como se puede observar se utilizan rutas para que estas sean mostradas en el puerto del localhost:8080.

Utilización de conexión a Mysql

En este apartado se utilizan varias clases o se puede decir que hay varias clases involucradas pero únicamente se describe la conexión directa que realiza python a la base de datos:

```
import jsonpickle
```

```

import mysql.connector

class ConexionDB:
    def insertarHorarios(self,horarios):
        connection = mysql.connector.connect(
            host="localhost",
            user="adminHorarios",
            password="password123",
            database="base_Horarios"
        )
        #Se se realiza a json pa guardarse
        serializedHorario1 = jsonpickle.encode(horarios,
unpicklable=True)
        print(serializedHorario1)
        cursor = connection.cursor()
        #insert_query = "INSERT INTO your_table (data) VALUES
(%s)"
        insert_query = "INSERT INTO horario (fecha_hora, nombre,
datos) VALUES (%s, %s, %s);"
        cursor.execute(insert_query,
(self.getFechaActual(),horarios.nombre,serializedHorario1))
        connection.commit()
        cursor.close()
        connection.close()
        print("insertando")

```

Se importa el módulo `mysql.connector` para interactuar con la base de datos MySQL y el módulo `jsonpickle` es únicamente para serializar la clase que contiene los horarios y así se guarde en la tupla de datos, dentro de la tabla de horario en la base de datos.

Esta clase y método están diseñados para insertar registros en la tabla horario de la base de datos base Horarios con información específica relacionada con fechas, nombres y datos serializados en formato JSON.

Guardado de variables

Las variables que se mencionaron anteriormente se guardan como objetos volátiles dentro de varias clases en el sistema. Aquí se muestra un poco de su información y contenido aunque no se explicara el funcionamiento de cada uno de los métodos que utilizan.

Control de datos

En primer lugar se tiene una clase control que es la que se encarga de crear y guardar todos los objetos, por lo que cuando se sube el archivo de configuración del usuario este lo busca y lo analiza.

Una parte importante de este control es la forma en que guarda los datos. Para esto se utilizan vectores que guardan todos los objetos, al ser volátiles no necesitan una gran base de datos para guardarse.

Archivo de subida de datos

Para facilitar el uso de python y no tener que realizar un sistema de lectura de archivos complejo, para cada una de las variables se decidió optar por que se utilice una clase de python como plantilla para que ahí se agreguen los datos. Un ejemplo de la plantilla que se debe utilizar es el siguiente.

```
import datetime
from classes.controlDatos import ControlDatos

class MockData:
    def __init__(self):
        self.control1 = None

    def analizarDatos(self):
        control = ControlDatos()
        """Creacion de carreras:Codigo, Nombre"""
        control.setCarrera(1,"Ing. Ciencias y Sistemas")
        control.setCarrera(2,"Ing. Civil")

        """Creacion de cursos Codigo, Nombre, Creditos, Semestre,
```

```

Duracion(horas la semana), Carrera, Cantidad de estudiantes"""
    control.setCurso(1011,"Matematica Basica 2", 5, 2, 3, 1,
10)
    control.setCurso(1010,"Matematica Basica 1", 5, 1, 3, 1,
10)

    """Darle prioridad a ciertos cursos, si estos no se les
agrega se pondra por defult una prioridad de 0"""
    control.setPrioridadCurso(1010,10)
    control.setPrioridadCurso(1011,9)

    """Creacion de salones: Codigo de Salon, Cantidad de
Asientos"""
    control.setSalon(1,10)
    control.setSalon(2,10)
    control.setSalon(3,10)
    control.setSalon(4,10)

    """Creacion de profesores: Nombre, Cursos que puede dar,
horario de contratacion, carrera"""
    control.setProfesor("Peter", [1011,1012,1013,1014,1015],
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
    control.setProfesor("Saul", [1012,1013],
[datetime.time(8,00,00), datetime.time(18,00,00)],2)

    return control

```

Modelo implementado

Creación de distribucion de períodos

Ahora bien una de las partes más importantes de la creación del horario es poder partir este horario en partes pequeñas u horas de trabajo, ya que de esta forma se va a distribuir el trabajo de cada uno de los salones y periodos, para esto se utiliza el siguiente método suponiendo algunas variables que el usuario podría ingresar como la duración del periodo y la hora de inicio y final de este periodo.

```

"""Definir el tiempo de los periodos en minutos"""
duracionPeriodo = 50

```

```

        """Definir la hora de inicio y final de todo el horario
        laboral"""
        horaInicialLaboral = datetime.strptime("7:00 AM", "%I:%M
%p")
        horaFinalLaboral = datetime.strptime("1:00 PM", "%I:%M
%p")

        """Definir el semestre en el que se está trabajando; 0 =
        par 1 = impar"""
        semestre = 1

        """Generar tabla de periodos/salon"""

        #Calcular la cantidad de periodos en el horario laboral
        self.listaPeriodos =
self.calcular_periodos(horaInicialLaboral,horaFinalLaboral,duracio
nPeriodo)

def calcular_periodos(self, inicio, fin, duracion_periodo):
    # Convertir las horas iniciales y finales a minutos
    inicioMinutos = (inicio.hour * 60) + inicio.minute
    finMinutos = (fin.hour * 60) + fin.minute

    # Calcular la duración total en minutos
    duracionTotal = finMinutos - inicioMinutos

    # Calcular la cantidad de periodos completos
    periodosCompletos = duracionTotal // duracion_periodo

    periodosLista = []
    for i in range(periodosCompletos):
        periodoInicio = inicio + timedelta(minutes=i *
duracion_periodo)
        periodoFin = inicio + timedelta(minutes=(i + 1) *
duracion_periodo)

        periodosLista.append((periodoInicio.strftime("%I:%M %p"),
periodoFin.strftime("%I:%M %p")))

    return periodosLista

```

De esta forma se tiene la distribución de los periodos según su hora de inicio y finalización, así como también la duración de los periodos en general.

Luego al tener esta cantidad de periodos distribuidos se puede crear un periodo según una hora y un salón asignado, por lo que se realiza una creación de periodos según la cantidad de salones que hay y por cada salón se hace una cantidad de periodos según la lista de distribución creada con anterioridad.

Analizador de horario

El analizador de horario es el corazón del sistema ya que en él se toman las decisiones para poder asignar los cursos a ciertos periodos tomando en cuenta las restricciones y variables que se han guardado.

El método que se realiza es que primero se escogen los cursos del semestre que se quiere ordenar, por lo que se separan con el siguiente método

```
def cursosAOrdenarPorSemestre(self,cursos,semestre):
    cursosDevolver = []
    for curso in cursos:
        if semestre == 0: #PAR
            if (curso.semestre % 2) == 0:
                cursosDevolver.append(curso)
        elif semestre == 1: #IMPAR
            if (curso.semestre % 2) != 0:
                cursosDevolver.append(curso)

    return cursosDevolver
```

Luego se escoge el modo en que se ordenarán los cursos, ya sea que estos se quieran ordenar por código ascendente o descendientemente, o ya sea por prioridad dada o la cantidad de estudiantes.

Una vez escogido este tipo pasan a ser ordenados según se ordenó, para esto se utiliza la opción sorted, el código utilizado para ordenarlos es el siguiente:

```
cursosOrdenadosAscendente = sorted(cursosEnSemestre, key=lambda x:
```

```
x.codigo, reverse=False)
cursosOrdenadosDescendente = sorted(cursosEnSemestre, key=lambda
x: x.codigo, reverse=True)
cursosOrdenadosPrioridad = sorted(cursosEnSemestre, key=lambda x:
x.prioridad, reverse=True)
cursosOrdenadosCantidadEstudiantes = sorted(cursosEnSemestre,
key=lambda x: x.cantidadEstudiantes, reverse=True)
```

Ahora bien, como ya se mencionaba con anterioridad los ordenamientos por prioridad y por cantidad de estudiantes pueden contener la misma prioridad o cantidad en varios cursos por lo que conocer cuál es la mejor opción de asignación primeriza, por lo que a cada uno de estos paquetes de cursos con prioridades y cantidades iguales se les aleatoriza su asignación, y el usuario tiene la capacidad de aleatorizar la cantidad de veces que quiera, pero de esta forma obtendrá varios horarios. La forma en que se aleatoriza es utilizando el siguiente código:

```
def aleatorizarCursosPrioridad(self,cursos):
    cursosDevolver = []
    # Diccionario para almacenar las listas agrupadas
    grupos = {}

    # Itera a través de los números
    # Itera a través de las personas
    for curso in cursos:
        prioridad = curso.prioridad
        # Si la edad ya está en el diccionario, agrega la
persona a la lista correspondiente
        if prioridad in grupos:
            grupos[prioridad].append(curso)
        else:
            # Si la edad no está en el diccionario, crea una
nueva lista con la persona
            grupos[prioridad] = [curso]

    # Ahora grupos contiene las listas de números iguales
    listas_agrupadas = list(grupos.values())
    # Junta los cursos de los grupos
    for lista in listas_agrupadas:
        random.shuffle(lista)
        for elemento in lista:
            cursosDevolver.append(elemento)
```

```
return cursosDevolver
```

El corazón de este sistema es la asignación del curso en algún periodo y la forma en que lo hace es la misma que el diagrama de procesos que se mencionó con anterioridad, primero para tener una lista de periodos nueva se realiza una copia de esta y así no cambiar los objetos que ya se tenían, luego para hacer una asignación más precisa se trata de realizar la asignación 3 veces, aunque cada vez tiene un objetivo distinto:

1. La primera asignación trata de asignar los cursos con profesores obligatorios
2. La segunda asignación trata de asignar con profesores optativos o más bien no optativos sino que tienen la capacidad de darlo pero no es su clase principal.
3. De último se trata de asignar cursos con profesores de la carrera, aunque estos no fueron asignados, sino que se realiza de esta forma para poder brindar apoyo o mejor calidad al horario.

El código que se utiliza para validar todas estas asignaciones y restricciones es el siguiente:

[illegible]

[illegible]

Utilizacion del profesor

El profesor juega un papel importante dentro de la asignación ya que de él depende prácticamente si lo puede dar a una hora determinada, ya sea porque tendrá algún traslape o sino que este no esté en su hora de trabajo. De parte del profesor se utilizan 2 métodos importantes para la separación de las restricciones mencionadas, los métodos son:

```
def asignarPeriodoUsado(self, horarioFinal ,periodo):
    horarioEncontrado = None
    periodoEncontrado = 0
    #validar que el profesor esta en hora de trabajo
    estaEnHoraLaboral =
self.calcularEstaDentroHoraDeContratacion(periodo,horarioFinal.dur
acionPeriodo)
    #TODO: Advertir que no esta en hora de trabajo
    if estaEnHoraLaboral == 1:
        for horariosUsado in self.horariosUsados:
            if horarioFinal == horariosUsado.horarioFinal:
                horarioEncontrado = horariosUsado
                listaHorasUsadas = []
                for periodoProfesor in horariosUsado.periodos:

listaHorasUsadas.append(periodoProfesor.idHora)
                    if periodoProfesor.idperiodo ==
periodoProfesor:
                        #No se puede insertar en este periodo
porque ya esta insertado en este
                            periodoEncontrado = 1

horarioFinal.agregarUnaAdvertencia(1,"No se logro asignar el
profesor: " + self.nombre + " por que ya tiene asignado este
periodo a esta hora: " + periodo.getHoraEnString(), 1)
                                return 0 #No se logro asignar el
periodo

                    if periodo.idHora in listaHorasUsadas:
                        #No se puede insertar en este periodo
porque ya tiene asignado un periodo en esta hora
                            periodoEncontrado = 1
                                horarioFinal.agregarUnaAdvertencia(1,"No
```



```

se logro asignar el profesor: " + self.nombre + " por que ya tiene
asignado un periodo a esta hora: " + periodo.getHoraEnString(), 1)
        return 1 #No se logro asignar el periodo
    else:
        return 1 # No se logro asignar el periodo porque no
esta en horario laboral

    if horarioEncontrado == None:
        horarioInsertar = HorarioProfesor(horarioFinal)
        horarioInsertar.addPeriodo(periodo)
        self.horariosUsados.append(horarioInsertar)
        return 2 #Se asigno el periodo con el horario
    else:
        if periodoEncontrado == 0:
            horarioEncontrado.addPeriodo(periodo)
            return 3 #Se asigno el periodo el periodo

# Si retorna en 1 es que si esta en hora
def
calcularEstaDentroHoraDeContratacion(self, periodo, duracionPeriodo)
:
    #Hora inicial del periodo debe estar dentro de la hora
inicial del profesor
    horaInicialProfesor = self.horarioContratacion.horaInicial
    horaFinalProfesor = self.horarioContratacion.horaFinal

    horaInicialPeriodo = periodo.horario.horaInicial.time()
    horaFinalPeriodo = periodo.horario.horaFinal.time()

    iniciaEnHora = 0
    terminaEnHora = 0
    if horaInicialProfesor <= horaInicialPeriodo:
        iniciaEnHora = 1
    if horaFinalProfesor >= horaFinalPeriodo:
        terminaEnHora = 1

    if iniciaEnHora == 1 and terminaEnHora == 1:
        return 1
    #Hora final debe estar dentro de la hora del profesor
    print("Calcular horario de trabajo del profesor")

```

Advertencias

En este proyecto se manejan varias advertencias se componen de varios tipos:

- [1]Normal

Si es una advertencia Normal es muy probable que se pueda arreglar con el paso del tiempo, o en otras iteraciones que estén por realizarse

- [2]Grave

Se es una advertencia grave, quiere decir que una acción que se suponía que tenía que suceder no sucedió pero podría tener arreglo

- [3]Irreparable

Si es tiene una advertencia de tipo irreparable, entonces nos indica que el código no puede llegar a resolver el problema

Las advertencias también se rigen por un tipo o lugar de asignación esto indica sobre dónde fue el problema, las secciones son:

- [1]Asignación de profesor
- [2]Asignación de curso
- [3]Asignación de periodo

Estas se mostrarán debajo de cada horario creado para facilitar la vista de cada uno, así como también identificar las advertencias de cada horario porque sino ocurriría confusión en cuanto a la aleatoriedad se trata.

Algunas advertencias como en la asignación de carreras no existentes saldrán de otra forma, por ejemplo si la carrera que se está asignando no existe entonces si se asigno el curso sin carrera este aparecera de color de fondo rojo con un codigo hexadecimal: #FF0000, esto con el objetivo de advertir al usuario que ese curso se asignó pero no pertenece a ninguna carrera.

Verificación del modelo

Para verificar que este modelo funciona se utiliza la siguiente entrada y se tratan de agregar, modificar y quitar ciertos parámetros para ver si realmente creaba un horario y de la forma que se espera.

Se piensa en verificar de manera sencilla, para que se pueda visualizar a simple vista los cambios que ocurren al realizar los horarios, se prueban escenarios simples para verificar cada uno de los estados y posibles errores y advertencias que debería mostrar.

Para verificar se utiliza la siguiente configuración para crear el horario:

- Hora de inicio de horario: 8:00 AM
- Hora de fin de horario: 10:00 AM
- Semestre: IMPAR
- Cantidad en minutos de la duración del periodo: 60 min

El archivo de carga simplificado que se usará para esta verificación es el siguiente:

```
"""Creación de carreras:Codigo, Nombre"""
    control.setCarrera(1,"Ing. Ciencias y Sistemas")

    """Creación de cursos Codigo, Nombre, Creditos, Semestre,
Duracion(horas la semana), Carrera, Cantidad de estudiantes"""
    control.setCurso(1012,"Curso 1", 5, 1, 3, 1, 10)
    control.setCurso(1013,"Curso 2", 5, 3, 3, 1, 10)
    control.setCurso(1014,"Curso 3", 5, 5, 3, 1, 10)
    control.setCurso(1015,"Curso 4", 5, 3, 3, 2, 10)
    control.setCurso(1016,"Curso 5", 5, 1, 3, 1, 10)

    """Creacion de salones:Codigo de Salon, Cantidad de
Asientos"""
    control.setSalon(1,10)
    control.setSalon(2,10)
    control.setSalon(3,10)

    """Creacion de profesores: Nombre, Cursos que puede dar,
horario de contratacion, carrera"""
    control.setProfesor("Peter", [1012,1013,1014,1015],
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
    control.setProfesor("Saul", [1012,1013],
[datetime.time(8,00,00), datetime.time(18,00,00)],2)

    """Agregar algun profesor especifico a los cursos si es
necesario"""
    #control.setProfesorFijoACurso("Saul",1012)
```

Resultado verificación 1

Tipo de generación: Ascendente

	1	2	3
	Asientos:10	Asientos:10	Asientos:10
('08:00 AM', '09:00 AM')	1012 Curso 1 Peter Alumnos: 10 Semestre: 1	1013 Curso 2 Saul Alumnos: 10 Semestre: 3	
('09:00 AM', '10:00 AM')	1014 Curso 3 Peter Alumnos: 10 Semestre: 5		

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Grave	1	No se logro asignar un profesor optativo al curso: Curso 4
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 4
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 4
Irreparable	3	No se logro asignar el curso: Curso 5

Eficiencia de Cursos Asignados: 0.6

Eficiencia de Usabilidad de periodos: 0.3333333333333333

Como se puede observar se obtuvo un horario muy pequeño y con muchas advertencias, tal y como se había esperado. Ya que logra asignar horarios sin traslape de horario de profesor, ni de semestre y de forma ascendente según su código que se ingresó.

También se puede observar que las advertencias logran su cometido, ya que advierten al usuario sobre las acciones que intento realizar el sistema pero no pudo completar, así como también el nivel de gravedad de esa advertencia, ya que si nos podemos dar cuenta advertencias normales no hace que pare por completo sino que trata de buscar una solución al problema, al igual que con las advertencias graves, con la diferencia que cada vez se acerca a

un problema que no puede resolver, y es cuando se cae a las advertencias de tipo irreparables, ahí es donde el sistema no logra asignar algún curso o periodo por alguna razón, y de ninguna forma encontrará una solución a ese problema.

La eficiencia aquí toma un papel importante ya que nos indica con números si se logró o no realizar una asignación completa, en este ejemplo como no se lograron asignar todos los cursos la eficiencia baja en cuestión, pero si se hubieran logrado asignar todos los cursos esta sería de 1.

Resultado verificación 2

Tipo de generación: Descendente

	1	2	3
	Asientos:10	Asientos:10	Asientos:10
('08:00 AM', '09:00 AM')	1015 Curso 4 Peter Alumnos: 10 Semestre: 3	1012 Curso 1 Saul Alumnos: 10 Semestre: 1	
('09:00 AM', '10:00 AM')	1014 Curso 3 Peter Alumnos: 10 Semestre: 5	1013 Curso 2 Saul Alumnos: 10 Semestre: 3	

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 2 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 5

Eficiencia de Cursos Asignados: 0.8

Eficiencia de Usabilidad de periodos: 0.5

Ahora bien como se puede observar la eficiencia del resultado a aumentado así como también los cursos que se asignaron, y es que esto sucede ya que estos cursos con un código mayor tienen un mayor codigo solo lo puede dar un profesor mientras que los de menor código los pueden dar los 2 profesores, y al asignarse primero los que solo lo pueden dar 1 profesor deja libre al otro para que pueda dar los otros cursos. Aunque al no tener suficientes profesores no se logran asignar más cursos ni se logra tener una mayor eficiencia.

Lo bueno es que sucede completamente lo mismo que en la anterior verificación, sobre las advertencias, y es que muestra lo que trata de hacer al asignar ciertos cursos pero al final no lo logra asignar debido a la falta de profesores.

Resultado verificación 3

Para esta verificación se agrega una prioridad a un curso, para que este se asigne siempre antes que los demás.

La prioridad que se asigna es la siguiente:

```
control.setPrioridadCurso(1014,10)
```

Por lo que indica que el curso con código 1014, que es el curso 3, se asignará antes que otros, mientras que los demás tendrán que hallar un espacio de manera aleatoria. Para esto se realizan 3 horarios distintos para poder ver cual es el mejor pero siempre tendría que asignarse el curso 3 de primero.

El resultado es el siguiente:

Horario #1

Eficiencia de Cursos Asignados: 0.8

Eficiencia de Usabilidad de periodos: 0.5

Anterior_Horario Siguiente_Horario

	1	2	3
	Asientos:10	Asientos:10	Asientos:10
('08:00 AM', '09:00 AM')	1014 Curso 3 Peter Alumnos: 10 Semestre: 5	1013 Curso 2 Saul Alumnos: 10 Semestre: 3	
('09:00 AM', '10:00 AM')	1015 Curso 4 Peter Alumnos: 10 Semestre: 3	1012 Curso 1 Saul Alumnos: 10 Semestre: 1	

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Saul por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Irreparable	3	No se logro asignar el curso: Curso 5

Horario#2

Eficiencia de Cursos Asignados: 0.6

Eficiencia de Usabilidad de periodos: 0.3333333333333333

[Anterior_Horario](#) [Siguiente_Horario](#)

		1	2	3
		Asientos:10	Asientos:10	Asientos:10
('08:00 AM', '09:00 AM')		1014 Curso 3 Peter Alumnos: 10 Semestre: 5	1012 Curso 1 Saul Alumnos: 10 Semestre: 1	
('09:00 AM', '10:00 AM')		1013 Curso 2 Peter Alumnos: 10 Semestre: 3		

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Saul por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 5
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Irreparable	3	No se logro asignar el curso: Curso 4
Irreparable	3	No se logro asignar el curso: Curso 5

Horario #3

Eficiencia de Cursos Asignados: 0.6

Eficiencia de Usabilidad de periodos: 0.3333333333333333

[Anterior_Horario](#) [Siguiente_Horario](#)

		1	2	3
		Asientos:10	Asientos:10	Asientos:10
('08:00 AM', '09:00 AM')		1014 Curso 3 Peter Alumnos: 10 Semestre: 5	1012 Curso 1 Saul Alumnos: 10 Semestre: 1	
('09:00 AM', '10:00 AM')		1015 Curso 4 Peter Alumnos: 10 Semestre: 3		

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Saul por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 2 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 2 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 2

Resultado verificación 4

A todos los cursos se le redujo a 9 la cantidad de alumnos.

Horario #1

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Grave	1	No se logro asignar un profesor optativo al curso: Curso 3
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 5
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 3
Irreparable	3	No se logro asignar el curso: Curso 3
Irreparable	3	No se looro asignar el curso: Curso 5

Horario #2

Eficiencia de Cursos Asignados: 0.6
 Eficiencia de Usabilidad de periodos: 0.3333333333333333
[Anterior_Horario](#) [Siguiente_Horario](#)

	1	2	3
	Asientos:10	Asientos:10	Asientos:10
('08:00 AM', '09:00 AM')	1012 Curso 1 Peter Alumnos: 10 Semestre: 1	1013 Curso 2 Saul Alumnos: 9 Semestre: 3	
('09:00 AM', '10:00 AM')	1014 Curso 3 Peter Alumnos: 9 Semestre: 5		

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Grave	1	No se logro asignar un profesor optativo al curso: Curso 4
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 5
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Irreparable	1	No se logro asignar un ningun profesor al curso: Curso 4
Irreparable	3	No se logro asignar el curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 4

A pesar de que solo se realizaron dos horarios se visualiza perfectamente en las advertencias que el orden de asignación es distinto a como se hace en cada uno de los horarios. Esto debido a la aleatoriedad que se le está prestando para crear la lista de asignación de los cursos.

Aunque algo que sí se puede visualizar es que el curso con más cantidad de alumnos es quien se asigna primero que todos.

Experimentación

Experimento #1

Este primer experimento es bastante simple, se tiene la idea de generar un horario de un solo salón con una sola hora de trabajo, con 5 cursos distintos, con la misma cantidad de estudiantes asignados. Esto con el objetivo de poder visualizar la asignación aleatoria de y con qué frecuencia es posible encontrar resultados iguales.

Para ello se considera la siguiente configuración:

- Hora de inicio de horario: 8:00 AM
- Hora de fin de horario: 9:00 AM
- Semestre: IMPAR
- Cantidad en minutos de la duración del periodo: 60 min

El archivo de configuración que se utilizó es el siguiente:

```
"""Creación de carreras:Codigo, Nombre"""
    control.setCarrera(1,"Ing. Ciencias y Sistemas")

    """Creacion de cursos Codigo, Nombre, Creditos, Semestre,
Duracion(horas la semana), Carrera, Cantidad de estudiantes"""
    control.setCurso(1011,"Curso 1", 5, 1, 3, 1, 10)
    control.setCurso(1012,"Curso 2", 5, 3, 3, 1, 10)
    control.setCurso(1013,"Curso 3", 5, 5, 3, 1, 10)
    control.setCurso(1014,"Curso 4", 5, 3, 3, 1, 10)
    control.setCurso(1015,"Curso 5", 5, 1, 3, 1, 10)

    """Creacion de salones:Codigo de Salon, Cantidad de
Asientos"""
    control.setSalon(1,10)

    """Creacion de profesores: Nombre, Cursos que puede dar,
horario de contratacion, carrera"""
    control.setProfesor("Peter", [1011,1012,1013,1014,1015],
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
```

El resultado que se obtuvo fueron los siguientes 5 horarios:

Horario 1

	1 Asientos:10
('08:00 AM', '09:00 AM')	1013 Curso 3 Peter Alumnos: 10 Semestre: 5

Advertencias de la creacion del horario

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Irreparable	3	No se logro asignar el curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 1
Irreparable	3	No se logro asignar el curso: Curso 2
Irreparable	3	No se logro asignar el curso: Curso 4

Horario 2

	1 Asientos:10
('08:00 AM', '09:00 AM')	1012 Curso 2 Peter Alumnos: 10 Semestre: 3

Advertencias de la creacion del horario

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Irreparable	3	No se logro asignar el curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 1
Irreparable	3	No se logro asignar el curso: Curso 4
Irreparable	3	No se logro asignar el curso: Curso 3

Horario 3

	1 Asientos:10
('08:00 AM', '09:00 AM')	1015 Curso 5 Peter Alumnos: 10 Semestre: 1

Advertencias de la creacion del horario

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Irreparable	3	No se logro asignar el curso: Curso 1
Irreparable	3	No se logro asignar el curso: Curso 2
Irreparable	3	No se logro asignar el curso: Curso 4
Irreparable	3	No se logro asignar el curso: Curso 3

Horario 4

	1
	Asientos:10
('08:00 AM', '09:00 AM')	1015 Curso 5 Peter Alumnos: 10 Semestre: 1

Advertencias de la creacion del horario

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Irreparable	3	No se logro asignar el curso: Curso 1
Irreparable	3	No se logro asignar el curso: Curso 2
Irreparable	3	No se logro asignar el curso: Curso 4
Irreparable	3	No se logro asignar el curso: Curso 3

Horario 5

	1
	Asientos:10
('08:00 AM', '09:00 AM')	1013 Curso 3 Peter Alumnos: 10 Semestre: 5

Advertencias de la creacion del horario

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Irreparable	3	No se logro asignar el curso: Curso 5
Irreparable	3	No se logro asignar el curso: Curso 1
Irreparable	3	No se logro asignar el curso: Curso 2
Irreparable	3	No se logro asignar el curso: Curso 4

Resultados obtenidos

Horario	Orden
1	3,5,1,2,4
2	2,5,1,4,3
3	5,1,2,4,3
4	5,1,2,4,3
5	3,5,1,2,4

Interpretación

Según los resultados obtenidos hay % probabilidades que se obtenga el mismo ordenamiento.

A pesar de que esto es muy difícil que ocurra según la documentación de random en python, es posible que suceda, por lo que confiar en que todas las veces se obtendrá un resultado aleatorio es dejarlo mucho a la suerte y hay que asegurarse de que se están realizando las suficientes pruebas para que esto no suceda seguido.

Una recomendación para realizar este tipo de horarios es que si se toma demasiado tiempo en realizar muchas corridas, entonces sería buena idea dividir esas corridas en partes, si son 50 entonces realizar y subir el mismo archivo de configuración unas 3 veces y realizar unos 20 horarios para así obtener una cantidad buena de horarios y una probabilidad menor de que se obtenga un ordenamiento similar, así como también un tiempo de espera menor. Tal vez hay que realizar un poco más de esfuerzo al ingresar varias veces la misma configuración pero podría convenir mejor si se van a obtener resultados más balanceados .

Experimento #2

La otra idea que se tiene es modificar un poco el experimento anterior y probar hasta qué cantidad de horarios creados es que logra tardar un poco más.

Por lo que se dispone a utilizar el mismo archivo de carga, pero con la diferencia de que ahora se usará la configuración por default del sistema por lo que se verá afectado el horario. Ya que ahora sí se podrán asignar todos los cursos en horarios, pero ahora es tomar en cuenta el tiempo que se tarda en cargar los horarios.

Configuración:

- Hora de inicio de horario: 7:00 AM
- Hora de fin de horario: 1:00 PM
- Semestre: IMPAR
- Cantidad en minutos de la duración del periodo: 60 min

Resultados Obtenidos

Cantidad de horarios	Tiempo transcurrido
1	Inmediato
10	Inmediato
50	Alrededor de medio segundo
100	2 segundos

1000	19 segundos
------	-------------

Interpretación de los resultados

Según los datos obtenidos realmente no tarda tanto en generar una gran cantidad de horarios donde solo exista un salón y las asignaciones son relativamente fáciles, ya que solo existe un profesor pero varios periodos y horarios en lo que podría entrar.

Se concluye lo siguiente sobre cada una de las iteraciones que se realizaron:

- Para una sola iteración (1 horario), el tiempo es "Inmediato". Esto sugiere que crear un solo horario es una tarea muy rápida y prácticamente instantánea.
- Cuando aumentas la cantidad de horarios a 10, todavía es "Inmediato", lo que indica que crear 10 horarios también es una tarea muy rápida y casi instantánea.
- A medida que aumenta la cantidad de horarios a 50, el tiempo se incrementa a medio segundo. Esto sugiere que la creación de 50 horarios lleva un poco más de tiempo en comparación con las cantidades más pequeñas, pero aún es una operación bastante rápida.
- Crear 100 horarios lleva más tiempo en comparación con cantidades más pequeñas, pero todavía es una operación manejable en términos de tiempo.
- Finalmente, cuando la cantidad de horarios aumenta a 1000, el tiempo transcurrido aumenta significativamente a "19 segundos". Esto sugiere que crear 1000 horarios es una tarea que lleva bastante tiempo en comparación con las cantidades más pequeñas.

En resumen, la tabla muestra que la creación de horarios se vuelve más lenta a medida que la cantidad de horarios aumenta. Es importante tener en cuenta que el tiempo necesario para completar esta tarea puede depender de varios factores, como la complejidad de los horarios, la eficiencia del software o algoritmo empleado para crear los horarios.

Recomendaciones a tomar en cuenta:

- Optimizar los procesos de creación: Si es posible, sería beneficioso buscar formas de optimizar el proceso de creación de horarios, especialmente cuando se trata de cantidades más grandes.
- Si la creación de 1000 horarios se realiza con frecuencia y es una parte importante de tu trabajo o negocio hay que considerar distribuir la carga en múltiples corridas o iteraciones. También habría que tomar en cuenta algún método de redistribución de carga. Para alcanzar una mejora en la escalabilidad horizontal.

Experimento #3

Este experimento consiste en realizar un horario un poco más grande de los que se hayan probado y visualizar cuáles podrían ser los fallos o reacciones con los distintos tipos de asignación que se tienen.

Se contempla la creación de 15 profesores, 3 carreras cada una con 3 profesores, un total de 40 cursos, en un horario de 7:00 AM a 1:00 PM.

Los profesores tendrán la capacidad de dar 5 cursos nada más, aparte si necesita la inserción de otro curso tomará el de la carrera pero en caso alguno solo tendrá esos 5. Y todos empezaran su hora de trabajo después de las 8:00 AM para que la primera hora siempre quede vacía.

Habrían 6 salones por lo que quedarían perfectos si la restricción del semestre no existiera, por lo que sí se tendrán periodos vacíos.

En este modelo se trata la manera de que el sistema logre asignar la mayoría de los cursos pero de manera distinta, de tal forma que pueda utilizar todos los recursos que se le brindaron y así poder dar varios horarios según los criterios de asignación que se brindan al usuario.

Además poder visualizar si la prioridad y aleatorización es una parte importante del sistema o no.

El código que se utiliza es el siguiente:

```
control.setCarrera(1,"Ing. Ciencias y Sistemas")
control.setCarrera(2,"Ing. Mecatronica")
control.setCarrera(5,"Ing. Industrial")

""Creacion de cursos Codigo, Nombre, Creditos, Semestre,
Duracion(horas la semana), Carrera, Cantidad de estudiantes""
control.setCurso(1011,"Curso 1", 5, 1, 3, 1, 10)
control.setCurso(1012,"Curso 2", 5, 3, 3, 1, 10)
control.setCurso(1013,"Curso 3", 5, 5, 3, 1, 10)
control.setCurso(1014,"Curso 4", 5, 7, 3, 1, 10)
control.setCurso(1015,"Curso 5", 5, 9, 3, 1, 10)
control.setCurso(1016,"Curso 6", 5, 1, 3, 1, 10)
control.setCurso(1017,"Curso 7", 5, 3, 3, 1, 10)
control.setCurso(1018,"Curso 8", 5, 5, 3, 1, 10)
control.setCurso(1019,"Curso 9", 5, 7, 3, 1, 10)
control.setCurso(1020,"Curso 10", 5, 9, 3, 1, 10)
control.setCurso(1021,"Curso 11", 5, 1, 3, 1, 10)
```

```
control.setCurso(1022,"Curso 12", 5, 3, 3, 1, 10)
control.setCurso(1023,"Curso 13", 5, 5, 3, 1, 10)
control.setCurso(1024,"Curso 14", 5, 7, 3, 1, 10)
control.setCurso(1025,"Curso 15", 5, 9, 3, 1, 10)
control.setCurso(1026,"Curso 16", 5, 1, 3, 2, 10)
control.setCurso(1027,"Curso 17", 5, 3, 3, 2, 10)
control.setCurso(1028,"Curso 18", 5, 5, 3, 2, 10)
control.setCurso(1029,"Curso 19", 5, 7, 3, 2, 10)
control.setCurso(1030,"Curso 20", 5, 9, 3, 2, 10)
control.setCurso(1031,"Curso 21", 5, 1, 3, 2, 10)
control.setCurso(1032,"Curso 22", 5, 3, 3, 2, 10)
control.setCurso(1033,"Curso 23", 5, 5, 3, 2, 10)
control.setCurso(1034,"Curso 24", 5, 7, 3, 2, 10)
control.setCurso(1035,"Curso 25", 5, 9, 3, 2, 10)
control.setCurso(1036,"Curso 26", 5, 1, 3, 2, 10)
control.setCurso(1037,"Curso 27", 5, 3, 3, 2, 10)
control.setCurso(1038,"Curso 28", 5, 5, 3, 2, 10)
control.setCurso(1039,"Curso 29", 5, 7, 3, 2, 10)
control.setCurso(1040,"Curso 30", 5, 9, 3, 2, 10)
control.setCurso(1041,"Curso 31", 5, 1, 3, 2, 10)
control.setCurso(1042,"Curso 32", 5, 3, 3, 2, 10)
control.setCurso(1043,"Curso 33", 5, 5, 3, 3, 10)
control.setCurso(1044,"Curso 34", 5, 7, 3, 3, 10)
control.setCurso(1045,"Curso 35", 5, 9, 3, 3, 10)
control.setCurso(1046,"Curso 36", 5, 1, 3, 3, 10)
control.setCurso(1047,"Curso 37", 5, 3, 3, 3, 10)
control.setCurso(1048,"Curso 38", 5, 5, 3, 3, 10)
control.setCurso(1049,"Curso 39", 5, 7, 3, 3, 10)
control.setCurso(1050,"Curso 40", 5, 9, 3, 3, 10)
```

```
"""Creacion de salones: Codigo de Salon, Cantidad de
Asientos"""
```

```
control.setSalon(1,10)
control.setSalon(2,10)
control.setSalon(3,10)
control.setSalon(4,10)
control.setSalon(5,10)
control.setSalon(6,10)
```

```
"""Creacion de profesores: Nombre, Cursos que puede dar,
horario de contratacion, carrera"""
```

```
control.setProfesor("Dolore", [1011,1012,1013],
```



```
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
    control.setProfesor("Maria1", [1014,1015,1016],
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
    control.setProfesor("Paulon", [1017,1018,1019],
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
    control.setProfesor("Wilson", [1020,1021,1022],
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
    control.setProfesor("Cinzia", [1023,1024,1025],
[datetime.time(8,00,00), datetime.time(18,00,00)],1)
    control.setProfesor("Aguado", [1026,1027,1028],
[datetime.time(8,00,00), datetime.time(18,00,00)],2)
    control.setProfesor("Bernad", [1029,1031,1032],
[datetime.time(8,00,00), datetime.time(18,00,00)],2)
    control.setProfesor("Blasco", [1033,1034,1035],
[datetime.time(8,00,00), datetime.time(18,00,00)],2)
    control.setProfesor("Boiras", [1036,1037,1038],
[datetime.time(8,00,00), datetime.time(18,00,00)],2)
    control.setProfesor("Ariase", [1039,1040,1041],
[datetime.time(8,00,00), datetime.time(18,00,00)],2)
    control.setProfesor("Alvaro", [1042,1043,1044],
[datetime.time(8,00,00), datetime.time(18,00,00)],3)
    control.setProfesor("Santos", [1045,1046,1047],
[datetime.time(8,00,00), datetime.time(18,00,00)],3)
    control.setProfesor("Mayrae", [1048,1049,1050],
[datetime.time(8,00,00), datetime.time(18,00,00)],3)
    control.setProfesor("Guidor", [1042,1043,1044],
[datetime.time(8,00,00), datetime.time(18,00,00)],3)
    control.setProfesor("Manuel", [1045,1046,1047],
[datetime.time(8,00,00), datetime.time(18,00,00)],3)
```

Horario #1

Tipo de asignación: Ascendente

Numero de Horario Generado: 0

Eficiencia de Cursos Asignados: 0.625

Eficiencia de Usabilidad de periodos: -0.363636363636365

[Anterior_Horario](#) [Siguiente_Horario](#)

	1	2	3	4	5	6
	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10
('07:00 AM', '08:00 AM')						
('08:00 AM', '09:00 AM')	1011 Curso 1 Dolore Alumnos: 10 Semestre: 1	1014 Curso 4 Marial Alumnos: 10 Semestre: 7	1017 Curso 7 Paulon Alumnos: 10 Semestre: 3	1020 Curso 10 Wilson Alumnos: 10 Semestre: 9	1023 Curso 13 Cinzia Alumnos: 10 Semestre: 5	
('09:00 AM', '10:00 AM')	1012 Curso 2 Dolore Alumnos: 10 Semestre: 3	1015 Curso 5 Marial Alumnos: 10 Semestre: 9	1018 Curso 8 Paulon Alumnos: 10 Semestre: 5	1021 Curso 11 Wilson Alumnos: 10 Semestre: 1	1024 Curso 14 Cinzia Alumnos: 10 Semestre: 7	
('10:00 AM', '11:00 AM')	1013 Curso 3 Dolore Alumnos: 10 Semestre: 5	1016 Curso 6 Marial Alumnos: 10 Semestre: 1	1019 Curso 9 Paulon Alumnos: 10 Semestre: 7	1022 Curso 12 Wilson Alumnos: 10 Semestre: 3	1025 Curso 15 Cinzia Alumnos: 10 Semestre: 9	
('11:00 AM', '12:00 PM')	1026 Curso 16 Aguado Alumnos: 10 Semestre: 1	1029 Curso 19 Bernad Alumnos: 10 Semestre: 7	1033 Curso 23 Blasco Alumnos: 10 Semestre: 5	1037 Curso 27 Boiras Alumnos: 10 Semestre: 3	1040 Curso 30 Ariase Alumnos: 10 Semestre: 9	
('12:00 PM', '01:00 PM')	1027 Curso 17 Aguado Alumnos: 10 Semestre: 3	1031 Curso 21 Bernad Alumnos: 10 Semestre: 1	1034 Curso 24 Blasco Alumnos: 10 Semestre: 7	1038 Curso 28 Boiras Alumnos: 10 Semestre: 5	1045 Curso 35 Santos Alumnos: 10 Semestre: 9	

Horario #2

Tipo de asignación: Descendente

Eficiencia de Cursos Asignados: 0.625

Eficiencia de Usabilidad de periodos: -0.363636363636365

[Anterior_Horario](#) [Siguiente_Horario](#)

	1	2	3	4	5	6
	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10
('07:00 AM', '08:00 AM')						
('08:00 AM', '09:00 AM')	1050 Curso 40 Mayrae Alumnos: 10 Semestre: 9	1047 Curso 37 Santos Alumnos: 10 Semestre: 3	1046 Curso 36 Manuel Alumnos: 10 Semestre: 1	1044 Curso 34 Alvaro Alumnos: 10 Semestre: 7	1043 Curso 33 Guidor Alumnos: 10 Semestre: 5	
('09:00 AM', '10:00 AM')	1049 Curso 39 Mayrae Alumnos: 10 Semestre: 7	1045 Curso 35 Santos Alumnos: 10 Semestre: 9	1042 Curso 32 Alvaro Alumnos: 10 Semestre: 3	1041 Curso 31 Ariase Alumnos: 10 Semestre: 1	1038 Curso 28 Boiras Alumnos: 10 Semestre: 5	
('10:00 AM', '11:00 AM')	1048 Curso 38 Mayrae Alumnos: 10 Semestre: 5	1040 Curso 30 Ariase Alumnos: 10 Semestre: 9	1037 Curso 27 Boiras Alumnos: 10 Semestre: 3	1034 Curso 24 Blasco Alumnos: 10 Semestre: 7	1031 Curso 21 Bernad Alumnos: 10 Semestre: 1	
('11:00 AM', '12:00 PM')	1039 Curso 29 Ariase Alumnos: 10 Semestre: 7	1036 Curso 26 Boiras Alumnos: 10 Semestre: 1	1035 Curso 25 Blasco Alumnos: 10 Semestre: 9	1032 Curso 22 Bernad Alumnos: 10 Semestre: 3	1028 Curso 18 Aguado Alumnos: 10 Semestre: 5	
('12:00 PM', '01:00 PM')	1033 Curso 23 Blasco Alumnos: 10 Semestre: 5	1029 Curso 19 Bernad Alumnos: 10 Semestre: 7	1027 Curso 17 Aguado Alumnos: 10 Semestre: 3	1025 Curso 15 Cinzia Alumnos: 10 Semestre: 9	1021 Curso 11 Wilson Alumnos: 10 Semestre: 1	

Horario #3

Tipo de asignación: Prioridad

Cantidad de corridas: 3

Aquí se toma en cuenta que todas las prioridades serán las mismas por lo que todos los cursos estarán asignados de forma aleatoria. De esta forma ya no hay necesidad de realizar otros horarios por cantidad de estudiantes porque igual se aleatoriza de la misma forma.

H1:

Eficiencia de Cursos Asignados: 0.625

Eficiencia de Usabilidad de periodos: -0.363636363636365

[Anterior_Horario](#) [Siguiente_Horario](#)

	1	2	3	4	5	6
	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10
('07:00 AM', '08:00 AM')						
('08:00 AM', '09:00 AM')	1023 Curso 13 Cinzia Alumnos: 10 Semestre: 5	1011 Curso 1 Dolore Alumnos: 10 Semestre: 1	1049 Curso 39 Mayrae Alumnos: 10 Semestre: 7	1015 Curso 5 Marial Alumnos: 10 Semestre: 9	1037 Curso 27 Boiras Alumnos: 10 Semestre: 3	
('09:00 AM', '10:00 AM')	1031 Curso 21 Bernad Alumnos: 10 Semestre: 1	1035 Curso 25 Blasco Alumnos: 10 Semestre: 9	1038 Curso 28 Boiras Alumnos: 10 Semestre: 5	1044 Curso 34 Alvaro Alumnos: 10 Semestre: 7	1012 Curso 2 Dolore Alumnos: 10 Semestre: 3	
('10:00 AM', '11:00 AM')	1021 Curso 11 Wilson Alumnos: 10 Semestre: 1	1039 Curso 29 Arlase Alumnos: 10 Semestre: 7	1032 Curso 22 Bernad Alumnos: 10 Semestre: 3	1013 Curso 3 Dolore Alumnos: 10 Semestre: 5	1045 Curso 35 Santos Alumnos: 10 Semestre: 9	
('11:00 AM', '12:00 PM')	1016 Curso 6 Marial Alumnos: 10 Semestre: 1	1029 Curso 19 Bernad Alumnos: 10 Semestre: 7	1020 Curso 10 Wilson Alumnos: 10 Semestre: 9	1033 Curso 23 Blasco Alumnos: 10 Semestre: 5	1042 Curso 32 Alvaro Alumnos: 10 Semestre: 3	
('12:00 PM', '01:00 PM')	1028 Curso 18 Arlase Alumnos: 10 Semestre: 5	1041 Curso 31 Arlase Alumnos: 10 Semestre: 1	1034 Curso 24 Blasco Alumnos: 10 Semestre: 7	1017 Curso 7 Paulon Alumnos: 10 Semestre: 3	1050 Curso 40 Mayrae Alumnos: 10 Semestre: 9	

H2:

Eficiencia de Cursos Asignados: 0.625

Eficiencia de Usabilidad de periodos: -0.363636363636365

[Anterior_Horario](#) [Siguiente_Horario](#)

	1	2	3	4	5	6
	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10
('07:00 AM', '08:00 AM')						
('08:00 AM', '09:00 AM')	1033 Curso 23 Bernad Alumnos: 10 Semestre: 5	1032 Curso 22 Marial Alumnos: 10 Semestre: 3	1015 Curso 5 Blasco Alumnos: 10 Semestre: 9	1046 Curso 36 Marial Alumnos: 10 Semestre: 1	1019 Curso 9 Paulon Alumnos: 10 Semestre: 7	
('09:00 AM', '10:00 AM')	1038 Curso 28 Boiras Alumnos: 10 Semestre: 5	1017 Curso 7 Paulon Alumnos: 10 Semestre: 3	1034 Curso 24 Blasco Alumnos: 10 Semestre: 7	1016 Curso 6 Marial Alumnos: 10 Semestre: 1	1025 Curso 15 Cinzia Alumnos: 10 Semestre: 9	
('10:00 AM', '11:00 AM')	1048 Curso 38 Mayrae Alumnos: 10 Semestre: 5	1012 Curso 2 Dolore Alumnos: 10 Semestre: 3	1021 Curso 11 Wilson Alumnos: 10 Semestre: 1	1029 Curso 19 Bernad Alumnos: 10 Semestre: 7	1035 Curso 25 Blasco Alumnos: 10 Semestre: 9	
('11:00 AM', '12:00 PM')	1037 Curso 27 Boiras Alumnos: 10 Semestre: 3	1023 Curso 13 Cinzia Alumnos: 10 Semestre: 5	1049 Curso 39 Mayrae Alumnos: 10 Semestre: 7	1011 Curso 1 Dolore Alumnos: 10 Semestre: 1	1020 Curso 10 Wilson Alumnos: 10 Semestre: 9	
('12:00 PM', '01:00 PM')	1027 Curso 17 Agudo Alumnos: 10 Semestre: 3	1039 Curso 29 Arlase Alumnos: 10 Semestre: 7	1031 Curso 21 Bernad Alumnos: 10 Semestre: 1	1050 Curso 40 Mayrae Alumnos: 10 Semestre: 9	1013 Curso 3 Dolore Alumnos: 10 Semestre: 5	

H3:

Eficiencia de Cursos Asignados: 0.625

Eficiencia de Usabilidad de periodos: -0.363636363636365

[Anterior_Horario](#) [Siguiente_Horario](#)

	1	2	3	4	5	6
	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10	Asientos:10
('07:00 AM', '08:00 AM')						
('08:00 AM', '09:00 AM')	1016 Curso 6 Marial Alumnos: 10 Semestre: 1	1037 Curso 27 Boiras Alumnos: 10 Semestre: 3	1033 Curso 23 Blasco Alumnos: 10 Semestre: 5	1045 Curso 35 Santos Alumnos: 10 Semestre: 9	1049 Curso 39 Mayrae Alumnos: 10 Semestre: 7	
('09:00 AM', '10:00 AM')	1047 Curso 37 Bernad Alumnos: 10 Semestre: 3	1031 Curso 21 Bernad Alumnos: 10 Semestre: 1	1013 Curso 3 Dolore Alumnos: 10 Semestre: 5	1050 Curso 40 Mayrae Alumnos: 10 Semestre: 9	1019 Curso 9 Paulon Alumnos: 10 Semestre: 7	
('10:00 AM', '11:00 AM')	1046 Curso 36 Santos Alumnos: 10 Semestre: 1	1048 Curso 38 Mayrae Alumnos: 10 Semestre: 5	1020 Curso 10 Wilson Alumnos: 10 Semestre: 9	1017 Curso 7 Paulon Alumnos: 10 Semestre: 3	1029 Curso 19 Bernad Alumnos: 10 Semestre: 7	
('11:00 AM', '12:00 PM')	1036 Curso 26 Boiras Alumnos: 10 Semestre: 1	1022 Curso 12 Wilson Alumnos: 10 Semestre: 3	1035 Curso 25 Blasco Alumnos: 10 Semestre: 9	1014 Curso 4 Marial Alumnos: 10 Semestre: 7	1028 Curso 18 Agudo Alumnos: 10 Semestre: 5	
('12:00 PM', '01:00 PM')	1026 Curso 16 Agudo Alumnos: 10 Semestre: 1	1044 Curso 34 Alvaro Alumnos: 10 Semestre: 7	1042 Curso 32 Guidor Alumnos: 10 Semestre: 3	1040 Curso 30 Arlase Alumnos: 10 Semestre: 9	1023 Curso 13 Cinzia Alumnos: 10 Semestre: 5	

Interpretación de los resultados

Como se puede observar en los primeros horarios creados, que fueron los de asignación ascendente y descendente realizan bien su trabajo, y tal como se predijo, el código valida que los profesores no están trabajando desde temprano por lo que no asigna a ningún profesor a esa hora, ni tampoco curso. En sí era más para poder visualizar si realizaba bien la distribución de los cursos aunque estos fueran de manera equitativa.

El sistema logró su cometido de probar al usuario que logra separar las restricciones impuestas anteriormente así como también presentar al usuario una aleatorización de manera continua.

En los últimos horarios que se crearon se realizaron 2 veces ya que el sistema no permitía guardar muchos horarios al mismo tiempo, por lo que se separaron 2 creaciones de horarios, y aunque se puede observar que los cursos presentan una forma aleatoria de asignación porque en ningún momento los horarios se repiten y si lo hacen hay mas de alguno que es distinto a alguna hora o en algun salon.

Lo más interesante de esta asignación es que a pesar de que se aleatorizan los cursos, de modo que ningún curso se va a asignar luego de otro, se puede observar que la eficiencia de todos los horarios es similar. Esto se debe a que la distribución de los profesores es bastante equitativa entonces el sistema puede asignar los cursos en los periodos sin ningún problema. El único inconveniente al realizar esto es que siempre se va a obtener la misma cantidad de cursos sin asignar ya que hay sobrepoblación de profesores y de espacio, pero no con las restricciones que se dan, ya sea por semestre o porque los profesores no pueden dar todos los cursos.

Manual de usuario

Como usuario hay que tener en cuenta varios requerimientos antes de utilizar el sistema.

Requisitos mínimos:

- Navegador Web
- Python 3
- Entorno de desarrollo virtual para alojar las librerías de python.
- Mysql o MariaDB

Librerías utilizadas:

- Datetime
- Copy
- Random
- matplotlib.pyplot : Ya no es necesario pero si da algun error es mejor instalarla
- bottle: Este es un micro web-framework, que permite la creación de un mini motor para poder visualizar html,css y js en un navegador web, así como el manejo de las rutas. En pocas palabras es un mini-flask.
- jsonpickle: Esta librería sirve para poder hacer una o varias clases en formato json y así poder guardarlas en la base de datos.
- mysql.connector

Hay que tener en cuenta que hay que realizar una instalación de Mysql y su base de datos, los comandos para realizar esta instalación con su debido usuario están contenidos en el apartado de Configuración de Base de datos.

Para iniciar el proyecto se deben de descargar todos los documentos y archivos que están alojados en el repositorio:

<https://github.com/waliray123/Proyecto1Modela2>

Una vez instalado todo se ejecuta el archivo initHost con python, en caso de utilizar un modelo virtualizado se realiza automáticamente.

La interfaz se ejecuta en el puerto de localhost numero 8080, por lo que en el navegador web hay que dirigirse a la dirección: <http://localhost:8080/>

De esta forma si se realizó todo bien se podrá visualizar la siguiente interfaz:

HORARIO GENERATIVO

Genera uno o varios horarios en base al archivo que se selecciono

Tipo de generacion

☐ CODIGO_ASCENDENTE

☐ CODIGO_DESCENDENTE

☐ PRIORIDAD

☐ CANTIDAD_ESTUDIANTES

Ingrese la hora inicial con el formato HH:MM AM/PM:

Ingrese la hora final con el formato HH:MM AM/PM:

Ingrese el semestre a generar PAR/IMPAR:

Ingrese la cantidad en minutos de la duracion del periodo:

Choose File

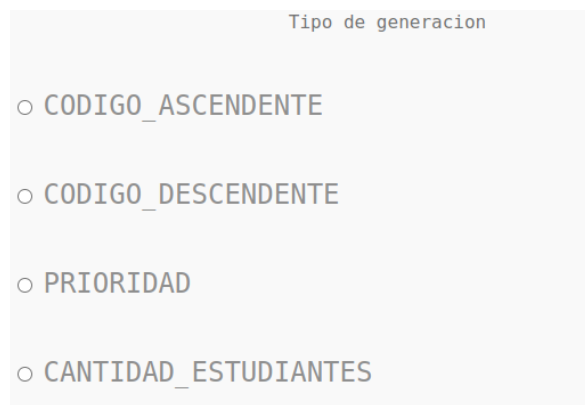
No file chosen

GENERAR

Hay que seleccionar las opciones que necesitemos y llenar los datos que nosotros convengamos, pero hay que asegurarse bien de qué es lo que se está ingresando porque el sistema tiene la configuración bastante sensible en el sentido de que tiene que ingresarse el formato que indica, si no tomara algunos datos erróneos o tomara los datos por default. Para generar los horarios no necesariamente hay que llenar todos los espacios, ya que si estos no se llenan por default se utilizan los siguientes:

- Hora de inicio de horario: 7:00 AM
- Hora de fin de horario: 1:00 PM
- Semestre: PAR
- Cantidad en minutos de la duración del periodo: 60 min

Para seleccionar el tipo de generación para ordenar los cursos se utilizan los radio buttons de letras más grandes, esto se realiza así con el objetivo de poder realizar solo 1 tipo de generación a la vez, y así evitar confusiones en la creación de los horarios.



Tipo de generacion

- ☐ CODIGO_ASCENDENTE
- ☐ CODIGO_DESCENDENTE
- ☐ PRIORIDAD
- ☐ CANTIDAD_ESTUDIANTES

Ahora bien, se ha mencionado que al utilizar la prioridad y la cantidad de estudiantes se tiene que especificar una cantidad de horarios que se generarán, se abrirá un nuevo texto donde se ingresara ese valor numérico para la generación de horarios así como en la siguiente imagen.



Tipo de generacion

- ☐ CODIGO_ASCENDENTE
- ☐ CODIGO_DESCENDENTE
- ☒ PRIORIDAD
- ☐ CANTIDAD_ESTUDIANTES

Ingrese la hora inicial con el formato HH:MM AM/PM:

Ingrese la hora final con el formato HH:MM AM/PM:

Ingrese el semestre a generar PAR/IMPAR:

Ingrese la cantidad en minutos de la duracion del periodo:

Ingrese un valor numérico:

Ahora bien hay que estar seguros de haber seleccionado el archivo de carga correcto, porque al no hacerlo el sistema reconocerá ese fallo y no logra generar ningún curso.

Una vez ingresado todos los datos o parámetros, se presiona el botón GENERAR

Tipo de generacion

☒ CODIGO_ASCENDENTE

☐ CODIGO_DESCENDENTE

☐ PRIORIDAD

☐ CANTIDAD_ESTUDIANTES

Ingrese la hora inicial con el formato HH:MM AM/PM:

Ingrese la hora final con el formato HH:MM AM/PM:

Ingrese el semestre a generar PAR/IMPAR:

Ingrese la cantidad en minutos de la duracion del periodo:

Choose File verificacion.py

GENERAR

Al presionar este botón se llevaran los datos al sistema, generará un horario y redirigirá a una ruta que mostrará el horario:

Horario ordenado segun cursos ordenados
ascendentemente

Numero de Horario Generado: 0

Eficiencia de Cursos Asignados: 0.6

Eficiencia de Usabilidad de periodos: 0.3333333333333333

[Anterior_Horario](#) [Siguiente_Horario](#)

	1	2	3
	Asientos:10	Asientos:10	Asientos:10
('08:00 AM', '09:00 AM')	1012 Curso 1 Peter Alumnos: 10 Semestre: 1	1013 Curso 2 Saul Alumnos: 9 Semestre: 3	
('09:00 AM', '10:00 AM')	1014 Curso BBB Peter Alumnos: 9 Semestre: 5		

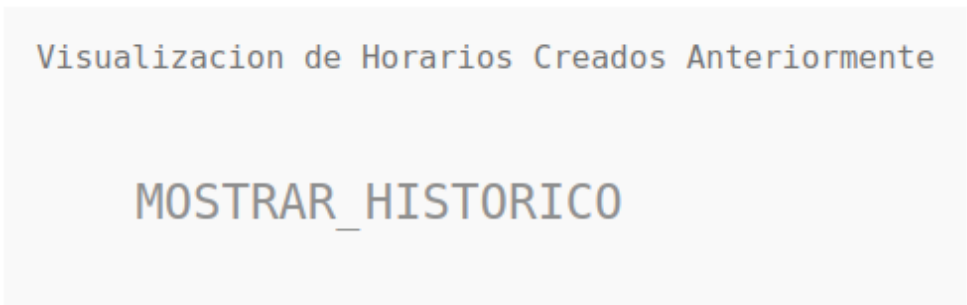
Además de mostrar el horario cabe aclarar que se pintaran los cursos de manera que se puedan diferenciar entre carreras.

A cada carrera se le asignará un color aleatorio al asignarse, y los cursos replicarán ese color en todo el horario.

Abajo del horario se podrán ver las advertencias que este tiro al estar realizando el horario, para saber el significado de estas advertencias dirigirse al apartado de Advertencias.

Tipo de Advertencia	Tipo de Asignacion	Tipo de Contenido
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 08:00:00 - 09:00:00
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Normal	1	No se logro asignar el profesor: Peter por que ya tiene asignado un periodo a esta hora: 09:00:00 - 10:00:00
Grave	1	No se logro asignar un profesor optativo al curso: Curso 4
Normal	2	No se logro asignar el curso: Curso 5 porque hay un periodo con un curso del mismo semestre asignado a esa hora
Grave	1	No se logro asignar un profesor optativo al curso: Curso 5
Normal	2	No se logro asignar el curso: Curso 4 porque hay un periodo con un curso del mismo semestre asignado a esa hora

Si se requiere visualizar un horario o varios que se han creado antes, entonces nos vamos al index y en el apartado de mostrar histórico hay un botón:



Se presiona este botón y se podrán visualizar todos los horarios que alguna vez se han creado, la visualización es como la siguiente imagen:

Historico de Horarios Creados			
ID Horario	Fecha Creacion	Nombre	Visualizacion
31	2023-09-17 17:06:18	Es una prueba de nombre	Redirigir
32	2023-09-17 17:13:39	Es una prueba de nombre	Redirigir
33	2023-09-17 17:13:59	Es una prueba de nombre	Redirigir
34	2023-09-17 17:14:24	Es una prueba de nombre	Redirigir
35	2023-09-17 17:14:32	Es una prueba de nombre	Redirigir
36	2023-09-17 17:14:52	Es una prueba de nombre	Redirigir

Si se requiere visualizar uno en concreto solo se selecciona el botón de redirigir y se mostrarán todos los horarios creados de ese tipo.

Historico de Horarios Creados

ID Horario	Fecha Creacion	Nombre	Visualizacion
11	2023-09-17 17:09:18	Es una prueba de nombre	Redirigir
12	2023-09-17 17:13:39	Es una prueba de nombre	Redirigir
13	2023-09-17 17:13:59	Es una prueba de nombre	Redirigir
14	2023-09-17 17:14:24	Es una prueba de nombre	Redirigir
15	2023-09-17 17:14:32	Es una prueba de nombre	Redirigir
16	2023-09-17 17:14:52	Es una prueba de nombre	Redirigir

Referencias

http://red.unal.edu.co/cursos/ciencias/2000352/html/un1/cont_116-16.html#:~:text=de%20dic ho%20efecto.-,La%20Aleatorizaci%C3%B3n,la%20estimaci%C3%B3n%20del%20error%20experimental.

<https://www.buscaminegocio.com/cursos-de-python/marco-web-hecho-en-python.html>

<https://idus.us.es/bitstream/handle/11441/115215/GM%20Walls%20Mestanza%2C%20Francisco.pdf?sequence=1&isAllowed=y>