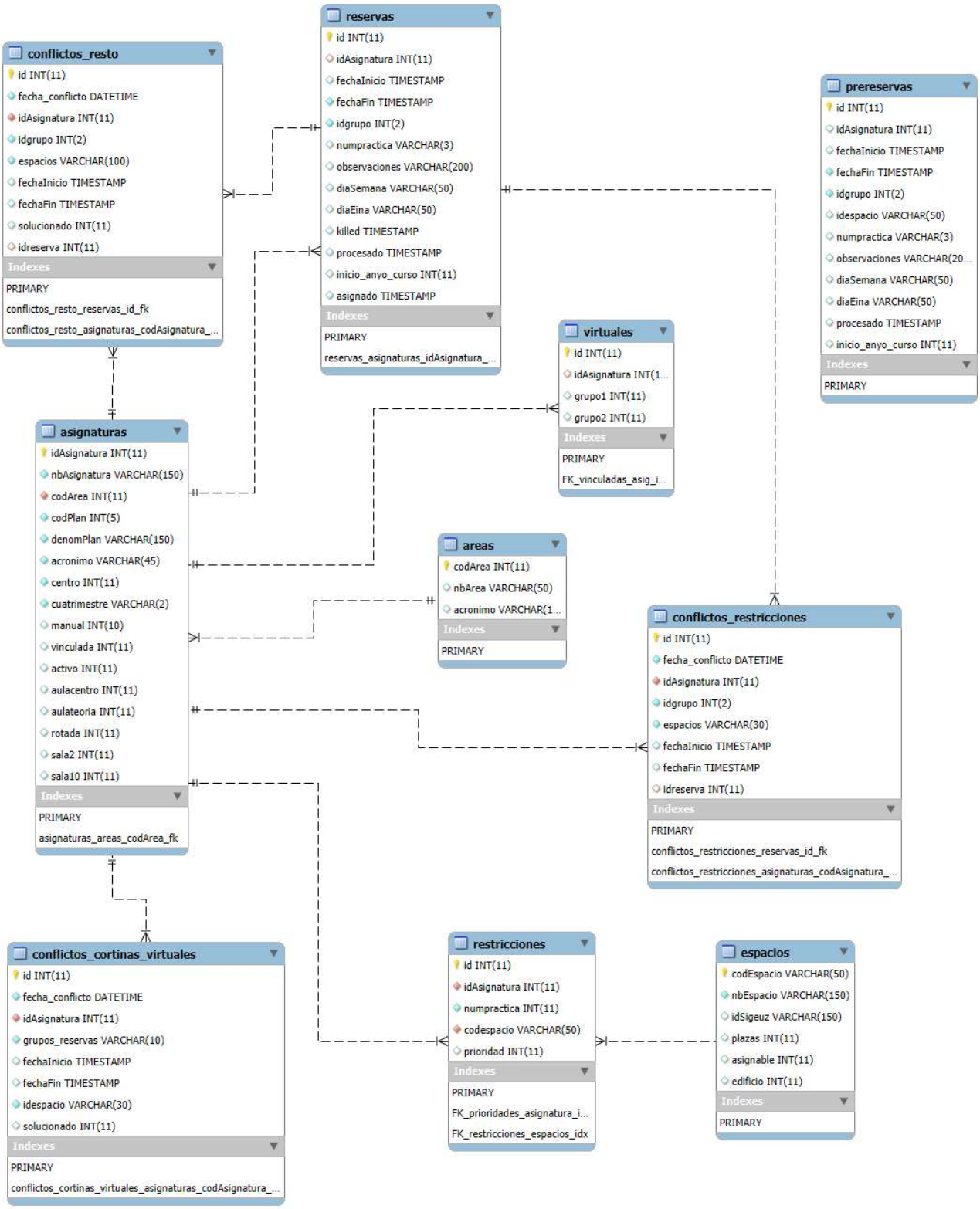
Esquema de Base de datos TAU



Script de creacion de tablas

```
-- --------------------------------------------------------
-- Host:                         danae03.unizar.es
-- Versión del servidor:         5.5.68-MariaDB - MariaDB Server
-- SO del servidor:              Linux
-- HeidiSQL Versión:             12.10.0.7000
-- --------------------------------------------------------

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

-- Volcando estructura para tabla tau.areas
CREATE TABLE IF NOT EXISTS `areas` (
  `codArea` int(11) NOT NULL DEFAULT '0' COMMENT 'Codigo de Area dentro de Odilo',
  `nbArea` varchar(50) DEFAULT NULL COMMENT 'Nombre del Area',
  `acronimo` varchar(15) DEFAULT NULL COMMENT 'Acronimo del Area',
  PRIMARY KEY (`codArea`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Tabla que contiene las areas que tienen asignaturas dentro del departamento.';


-- Volcando estructura para tabla tau.asignaturas
CREATE TABLE IF NOT EXISTS `asignaturas` (
  `idAsignatura` int(11) NOT NULL COMMENT 'Identificador de Asignatura',
  `nbAsignatura` varchar(150) CHARACTER SET utf8mb4 NOT NULL COMMENT 'Nombre de Asignatura',
  `codArea` int(11) NOT NULL COMMENT 'Area a la que pertenece la Asignatura',
  `codPlan` int(5) unsigned NOT NULL DEFAULT '0' COMMENT 'Codigo de plan al que pertenece la asignatura',
  `denomPlan` varchar(150) NOT NULL COMMENT 'Nombre del plan alque pertenece la asignatura',
  `acronimo` varchar(45) CHARACTER SET utf8mb4 NOT NULL COMMENT 'Acronimo de la asignatura compuesto por las primeras letras de cada palabra del nombre de la asignatura ''_'' primeras letras de cada palabra del nombre de la titulacion',
  `centro` int(11) unsigned NOT NULL COMMENT 'Codigo del centro al que pertenece la asignatura, 110',
  `cuatrimestre` varchar(2) NOT NULL COMMENT 'Cuatrimestre en el que se imparte la asignatura, S1 primer semestre, S2 segundo semestre, S rotadas\n',
  `manual` int(10) unsigned DEFAULT '0' COMMENT 'Marca si la asignatura se va a gestionar automatica o manual 0/1, por defecto autonatica 0\n',
  `vinculada` int(11) unsigned DEFAULT '0' COMMENT 'Si lo esta, codigo de la asignatura a la que esta vinculada',
  `activo` int(11) unsigned DEFAULT '1' COMMENT 'Campo que marca si la asignatura esta activa para que la procese el algoritmo, activo=1 ',
  `aulacentro` int(11) unsigned DEFAULT '0' COMMENT 'Campo que indica si la asignatura se imparte en aulas de centro, centro=1 ',
  `aulateoria` int(11) unsigned DEFAULT '0' COMMENT 'Campo que indica si la asignatura se imparte en el aula de teoria, teoria=1',
```

```
  `rotada` int(11) unsigned DEFAULT '0' COMMENT 'Campo que indica si la asignatura ha sido rotada, 0 no rotada, 1 rotada semestre 1, 2 rotada semestre 2',
  `sala2` int(11) unsigned DEFAULT '0' COMMENT 'Campo que indica si la asignatura se imparte en sala 2 del edificio Torres Quevedo, sala2=1',
  `sala10` int(11) unsigned DEFAULT '0' COMMENT 'Campo que indica si la asignatura se imparte en la sala 10 del edificio Torres Quevedo, sala10 = 1',
  PRIMARY KEY (`idAsignatura`) USING BTREE,
  KEY `asignaturas_areas_codArea_fk` (`codArea`),
  CONSTRAINT `asignaturas_areas_codArea_fk` FOREIGN KEY (`codArea`) REFERENCES `areas` (`codArea`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Tabla que contiene las asignaturas que pueden realizar practicas en DIIS';

-- Volcando estructura para tabla tau.conflictos_cortinas_virtuales
CREATE TABLE IF NOT EXISTS `conflictos_cortinas_virtuales` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fecha_conflicto` datetime NOT NULL COMMENT 'Fecha en la que se generó el conflicto',
  `idAsignatura` int(11) NOT NULL COMMENT 'Identificador de la Asignatura que provoca conflicto',
  `grupos_reservas` varchar(10) NOT NULL COMMENT 'Identificadores de los grupos de la asignatura separados por , que han intentado asignarse',
  `fechaInicio` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora de inicio de la reserva',
  `fechaFin` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora de fin de la reserva',
  `idespacio` varchar(30) NOT NULL COMMENT 'Identificador del espacio que quieren ocupar.',
  `solucionado` int(11) DEFAULT NULL COMMENT 'Indicador de si el conflicto esta solucionado o no, solucionado = 1',
  PRIMARY KEY (`id`),
  KEY `conflictos_cortinas_virtuales_asignaturas_codAsignatura_fk` (`idAsignatura`) USING BTREE,
  CONSTRAINT `conflictos_cortinas_virtuales_asignaturas_idAsignatura_fk` FOREIGN KEY (`idAsignatura`) REFERENCES `asignaturas` (`idAsignatura`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8 COMMENT='Tabla que almacena datos de conflictos de asignación de espacios a los grupos de cortinas virtuales';

-- Volcando estructura para tabla tau.conflictos_resto
CREATE TABLE IF NOT EXISTS `conflictos_resto` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fecha_conflicto` datetime NOT NULL COMMENT 'Fecha en la que se generó el conflicto',
  `idAsignatura` int(11) NOT NULL COMMENT 'Identificador de la Asignatura que provoca conflicto',
  `idgrupo` int(2) NOT NULL COMMENT 'Identificador del grupo de la asignatura que ha intentado asignarse',
  `espacios` varchar(100) NOT NULL COMMENT 'Resto de conflictos generados de reservas no privilegiadas\n',
  `fechaInicio` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora de inicio de la reserva',
  `fechaFin` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora de fin de la reserva',
  `solucionado` int(11) DEFAULT NULL COMMENT 'Indicador de si el conflicto esta solucionado o no, solucionado = 1',
  `idreserva` int(11) DEFAULT NULL COMMENT 'Clave foránea relacionada con la tabla reservas',
  PRIMARY KEY (`id`),
  KEY `conflictos_resto_reservas_id_fk` (`idreserva`),
  KEY `conflictos_resto_asignaturas_codAsignatura_fk` (`idAsignatura`) USING BTREE,
  CONSTRAINT `conflictos_resto_asignaturas_idAsignatura_fk` FOREIGN KEY (`idAsignatura`) REFERENCES `asignaturas` (`idAsignatura`) ON DELETE CASCADE,
  CONSTRAINT `conflictos_resto_reservas_id_fk` FOREIGN KEY (`idreserva`) REFERENCES `reservas` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=68 DEFAULT CHARSET=utf8 COMMENT='Tabla que almacena datos de conflictos de asignación de espacios al resto de asignaturas';
```

```
-- Volcando estructura para tabla tau.conflictos_restricciones
CREATE TABLE IF NOT EXISTS `conflictos_restricciones` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `fecha_conflicto` datetime NOT NULL COMMENT 'Fecha en la que se generó el conflicto',
  `idAsignatura` int(11) NOT NULL COMMENT 'Identificador de la Asignatura que provoca conflicto',
  `idgrupo` int(2) NOT NULL COMMENT 'Identificador del grupo de la asignatura que ha intentado asignarse',
  `espacios` varchar(30) NOT NULL COMMENT 'Conflictos generados de reservas restrictivas\n',
  `fechaInicio` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora de inicio de la reserva',
  `fechaFin` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora de fin de la reserva',
  `idreserva` int(11) DEFAULT NULL COMMENT 'Clave foránea relacionada con la tabla Reservas',
  PRIMARY KEY (`id`),
  KEY `conflictos_restricciones_reservas_id_fk` (`idreserva`),
  KEY `conflictos_restricciones_asignaturas_codAsignatura_fk` (`idAsignatura`) USING BTREE,
  CONSTRAINT `conflictos_restricciones_asignaturas_idAsignatura_fk` FOREIGN KEY (`idAsignatura`)
REFERENCES `asignaturas` (`idAsignatura`) ON DELETE CASCADE,
  CONSTRAINT `conflictos_restricciones_reservas_id_fk` FOREIGN KEY (`idreserva`) REFERENCES `reservas`
(`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8 COMMENT='Tabla que almacena datos de
conflictos de asignación de espacios a los grupos de restricciones';

-- Volcando estructura para vista tau.cortinas_reservas_sigma
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `cortinas_reservas_sigma` (
	`IDASIGNATURA` INT(11) NULL COMMENT 'Codigo de asignatura',
	`fechaInicio` TIMESTAMP NULL COMMENT 'TimeStamp con fecha y hora de inicio de la practica',
	`Hora_Par_Impar` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci',
	`fechaFin` TIMESTAMP NOT NULL COMMENT 'TimeStamp con fecha y hora de fin de la practica',
	`Duracion_Horas` DECIMAL(24,4) NULL,
	`idespacio` VARCHAR(1) NULL COMMENT 'Identificador del espacio donde se realiza la practica'
COLLATE 'utf8mb4_general_ci',
	`Grupo1` INT(2) NULL COMMENT 'Identificador del grupo de practicas',
	`Grupo2` INT(2) NULL COMMENT 'Identificador del grupo de practicas',
	`IDGRUPOS` TEXT NULL COLLATE 'utf8mb4_general_ci',
	`Total_Grupos` BIGINT(21) NOT NULL,
	`procesado` TIMESTAMP NULL COMMENT 'Fecha y hora en que se carga la reserva desde el json de
sigma',
	`asignado` TIMESTAMP NULL COMMENT 'Indica el momento en que se ha asignado espacio, echa y
hora'
) ENGINE=MyISAM;

-- Volcando estructura para tabla tau.espacios
CREATE TABLE IF NOT EXISTS `espacios` (
  `codEspacio` varchar(50) NOT NULL DEFAULT '' COMMENT 'Codigo del espacio L0.01, L0.02....',
  `nbEspacio` varchar(150) NOT NULL DEFAULT '' COMMENT 'Nombre del espacio, ej: Laboratorio 1,
Laboratorio 2...',
  `idSigeuz` varchar(150) DEFAULT '' COMMENT 'Identificador en SIGEUZ ej: CRE.1200.00.180....',
  `plazas` int(11) DEFAULT '0' COMMENT 'Numero de plazas que tiene el espacio',
  `asignable` int(11) DEFAULT '0' COMMENT 'Indica si es un espacio que podemos asignao ocupacion o no,
0/1 no asignable=1',
  `edificio` int(11) DEFAULT NULL COMMENT 'Identificador del edificio donde se encuentra el espacio',
  PRIMARY KEY (`codEspacio`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='Espacios en los que se pueden realizar practicas en DIIS';

-- Volcando estructura para tabla tau.prereservas
CREATE TABLE IF NOT EXISTS `prereservas` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Identificador Reserva',
  `idAsignatura` int(11) DEFAULT NULL COMMENT 'Identificador de asignatura',
  `fechaInicio` timestamp NULL DEFAULT NULL COMMENT 'TimeStamp con fecha de inicio de la practica',
  `fechaFin` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT 'TimeStamp con fecha de fin de la practica',
  `idgrupo` int(2) NOT NULL DEFAULT '0' COMMENT 'Identificador del grupo de practicas',
  `idespacio` varchar(50) DEFAULT NULL COMMENT 'Identificador de espacio',
  `numpractica` varchar(3) DEFAULT NULL COMMENT 'Numero de practica que corresponde a la reserva',
  `observaciones` varchar(200) DEFAULT NULL COMMENT 'Observaciones sobre la reserva.',
  `diaSemana` varchar(50) DEFAULT NULL COMMENT 'Lunes, martes....',
  `diaEina` varchar(50) DEFAULT NULL COMMENT 'Dia dentro del calendario de EINA, la1, lb1....',
  `procesado` timestamp NULL DEFAULT NULL COMMENT 'Dia en que es procesado del json de sigma',
  `inicio_anyo_curso` int(11) DEFAULT NULL COMMENT 'Inicio año del curso, 2025, 2026...',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=3566 DEFAULT CHARSET=utf8mb4 COMMENT='Tabla que contiene las reservas de espacios en DIIS utilizada para la sincronizacion';


-- Volcando estructura para tabla tau.reservas
CREATE TABLE IF NOT EXISTS `reservas` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Identificador Reserva',
  `idAsignatura` int(11) DEFAULT NULL COMMENT 'Codigo de asignatura',
  `fechaInicio` timestamp NULL DEFAULT NULL COMMENT 'TimeStamp con fecha y hora de inicio de la practica',
  `fechaFin` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' COMMENT 'TimeStamp con fecha y hora de fin de la practica',
  `idgrupo` int(2) NOT NULL DEFAULT '0' COMMENT 'Identificador del grupo de practicas',
  `idespacio` varchar(50) DEFAULT NULL COMMENT 'Identificador del espacio donde se realiza la practica',
  `numpractica` varchar(3) DEFAULT NULL COMMENT 'Numero de practica que corresponde a la reserva, P1, P2....',
  `observaciones` varchar(200) DEFAULT NULL COMMENT 'Observaciones sobre la reserva',
  `diaSemana` varchar(50) DEFAULT NULL COMMENT 'Lunes, martes, miercoles....',
  `diaEina` varchar(50) DEFAULT NULL COMMENT 'Dia dentro del calendario de EINA, la1, lb1....',
  `killed` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora en el que la reserva deja de tener validez',
  `procesado` timestamp NULL DEFAULT NULL COMMENT 'Fecha y hora en que se carga la reserva desde el json de sigma',
  `inicio_anyo_curso` int(11) DEFAULT NULL COMMENT 'Inicio año del curso, 2025, 2026...',
  `asignado` timestamp NULL DEFAULT NULL COMMENT 'Indica el momento en que se ha asignado espacio, echa y hora',
  PRIMARY KEY (`id`) USING BTREE,
  KEY `reservas_espacios_codEspacio_fk` (`idespacio`) USING BTREE,
  KEY `reservas_asignaturas_idAsignatura_fk` (`idAsignatura`),
  CONSTRAINT `reservas_asignaturas_idAsignatura_fk` FOREIGN KEY (`idAsignatura`) REFERENCES `asignaturas` (`idAsignatura`) ON DELETE CASCADE,
  CONSTRAINT `reservas_espacios_codEspacio_fk` FOREIGN KEY (`idespacio`) REFERENCES `espacios` (`codEspacio`)
) ENGINE=InnoDB AUTO_INCREMENT=223922 DEFAULT CHARSET=utf8mb4 COMMENT='Tabla que contiene las reservas de espacios en DIIS para la realización de las practicas de las asignaturas';
```

```
-- Volcando estructura para vista tau.resto_reservas_sigma
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `resto_reservas_sigma` (
	`id` INT(11) NOT NULL COMMENT 'Identificador Reserva',
	`IDASIGNATURA` INT(11) NULL COMMENT 'Codigo de asignatura',
	`IDGRUPO` INT(2) NOT NULL COMMENT 'Identificador del grupo de practicas',
	`fechainicio` TIMESTAMP NULL COMMENT 'TimeStamp con fecha y hora de inicio de la practica',
	`Hora_Par_Impar` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci',
	`fechafin` TIMESTAMP NOT NULL COMMENT 'TimeStamp con fecha y hora de fin de la practica',
	`Duracion_Horas` DECIMAL(24,4) NULL,
	`IDESPACIO` VARCHAR(1) NULL COMMENT 'Identificador del espacio donde se realiza la practica' COLLATE 'utf8mb4_general_ci',
	`NUMPRACTICA` VARCHAR(1) NULL COMMENT 'Numero de practica que corresponde a la reserva, P1, P2....' COLLATE 'utf8mb4_general_ci',
	`OBSERVACIONES` VARCHAR(1) NULL COMMENT 'Observaciones sobre la reserva' COLLATE 'utf8mb4_general_ci',
	`asignado` TIMESTAMP NULL COMMENT 'Indica el momento en que se ha asignado espacio, echa y hora'
) ENGINE=MyISAM;

-- Volcando estructura para tabla tau.restricciones
CREATE TABLE IF NOT EXISTS `restricciones` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Identificador autonumerico',
  `idAsignatura` int(11) NOT NULL COMMENT 'Identificador de la asignatura ',
  `numpractica` int(11) NOT NULL COMMENT 'Identificador del numero de practica que tiene la restricción',
  `codespacio` varchar(50) NOT NULL COMMENT 'Código del espacio que es preferente para esa practica',
  `prioridad` int(11) DEFAULT NULL COMMENT 'Indica la prioridad a tomar segun laboratorio 1, 2 y 3',
  PRIMARY KEY (`id`),
  KEY `FK_prioridades_asignatura_idx` (`idAsignatura`),
  KEY `FK_restricciones_espacios_idx` (`codespacio`) USING BTREE,
  CONSTRAINT `restricciones_asignaturas_idAsignatura_fk` FOREIGN KEY (`idAsignatura`) REFERENCES `asignaturas` (`idAsignatura`),
  CONSTRAINT `restricciones_espacios_codEspacio_fk` FOREIGN KEY (`codespacio`) REFERENCES `espacios` (`codEspacio`)
) ENGINE=InnoDB AUTO_INCREMENT=72 DEFAULT CHARSET=utf8mb4 COMMENT='Tabla que contiene las prioridades/restricciones a tener en cuenta a la hora de asignar los espacios, si no se incluye id de practica lo que se entendera es que es toda la asignatura.';

-- Volcando estructura para vista tau.restricciones_para_los_que_tienen_solo_dos_espacios
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `restricciones_para_los_que_tienen_solo_dos_espacios` (
	`idAsignatura` INT(11) NOT NULL COMMENT 'Identificador de la asignatura ',
	`numpractica` INT(11) NOT NULL COMMENT 'Identificador del numero de practica que tiene la restricción',
	`opcion1` VARCHAR(1) NULL COMMENT 'Código del espacio que es preferente para esa practica' COLLATE 'utf8mb4_general_ci',
	`opcion2` VARCHAR(1) NULL COMMENT 'Código del espacio que es preferente para esa practica' COLLATE 'utf8mb4_general_ci'
) ENGINE=MyISAM;
```

```
-- Volcando estructura para vista tau.restricciones_para_los_que_tienen_solo_tres_espacios
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `restricciones_para_los_que_tienen_solo_tres_espacios` (
	`idAsignatura` INT(11) NOT NULL COMMENT 'Identificador de la asignatura ',
	`numpractica` INT(11) NOT NULL COMMENT 'Identificador del numero de practica que tiene la restricción',
	`opcion1` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci',
	`opcion2` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci',
	`opcion3` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci'
) ENGINE=MyISAM;

-- Volcando estructura para vista tau.restricciones_para_los_que_tienen_solo_un_espacio
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `restricciones_para_los_que_tienen_solo_un_espacio` (
	`idAsignatura` INT(11) NOT NULL COMMENT 'Identificador de la asignatura ',
	`numpractica` INT(11) NOT NULL COMMENT 'Identificador del numero de practica que tiene la restricción'
) ENGINE=MyISAM;

-- Volcando estructura para vista tau.restricciones_reservas_de_solo_dos_espacios
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `restricciones_reservas_de_solo_dos_espacios` (
	`idAsignatura` INT(11) NOT NULL COMMENT 'Identificador de la asignatura ',
	`manual` INT(10) UNSIGNED NULL COMMENT 'Marca si la asignatura se va a gestionar automatica o manual 0/1, por defecto autonatica 0\n',
	`activo` INT(11) UNSIGNED NULL COMMENT 'Campo que marca si la asignatura esta activa para que la procese el algoritmo, activo=1 ',
	`numpractica` INT(11) NOT NULL COMMENT 'Identificador del numero de practica que tiene la restricción',
	`opcion1` VARCHAR(1) NULL COMMENT 'Código del espacio que es preferente para esa practica' COLLATE 'utf8mb4_general_ci',
	`opcion2` VARCHAR(1) NULL COMMENT 'Código del espacio que es preferente para esa practica' COLLATE 'utf8mb4_general_ci'
) ENGINE=MyISAM;

-- Volcando estructura para vista tau.restricciones_reservas_de_solo_tres_espacios
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `restricciones_reservas_de_solo_tres_espacios` (
	`idAsignatura` INT(11) NOT NULL COMMENT 'Identificador de la asignatura ',
	`manual` INT(10) UNSIGNED NULL COMMENT 'Marca si la asignatura se va a gestionar automatica o manual 0/1, por defecto autonatica 0\n',
	`activo` INT(11) UNSIGNED NULL COMMENT 'Campo que marca si la asignatura esta activa para que la procese el algoritmo, activo=1 ',
	`numpractica` INT(11) NOT NULL COMMENT 'Identificador del numero de practica que tiene la restricción',
	`opcion1` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci',
	`opcion2` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci',
	`opcion3` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci'
) ENGINE=MyISAM;
```

```
-- Volcando estructura para vista tau.restricciones_reservas_de_un_solo_espacio
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `restricciones_reservas_de_un_solo_espacio` (
	`idAsignatura` INT(11) NOT NULL COMMENT 'Identificador de la asignatura ',
	`manual` INT(10) UNSIGNED NULL COMMENT 'Marca si la asignatura se va a gestionar automatica o manual 0/1, por defecto autonatica 0\n',
	`activo` INT(11) UNSIGNED NULL COMMENT 'Campo que marca si la asignatura esta activa para que la procese el algoritmo, activo=1 ',
	`numpractica` INT(11) NOT NULL COMMENT 'Identificador del numero de practica que tiene la restricción',
	`codespacio` VARCHAR(1) NOT NULL COMMENT 'Código del espacio que es preferente para esa practica' COLLATE 'utf8mb4_general_ci',
	`prioridad` INT(11) NULL COMMENT 'Indica la prioridad a tomar segun laboratorio 1, 2 y 3'
) ENGINE=MyISAM;

-- Volcando estructura para vista tau.Total_CV_coincidentes_Reservas
-- Creando tabla temporal para superar errores de dependencia de VIEW
CREATE TABLE `Total_CV_coincidentes_Reservas` (
	`idAsignatura_Reservas` INT(11) NULL COMMENT 'Codigo de asignatura',
	`MANUAL` INT(10) UNSIGNED NULL COMMENT 'Marca si la asignatura se va a gestionar automatica o manual 0/1, por defecto autonatica 0\n',
	`ACTIVO` INT(11) UNSIGNED NULL COMMENT 'Campo que marca si la asignatura esta activa para que la procese el algoritmo, activo=1 ',
	`fechainicio` TIMESTAMP NULL COMMENT 'TimeStamp con fecha y hora de inicio de la practica',
	`Hora_Par_Impar` VARCHAR(1) NULL COLLATE 'utf8mb4_general_ci',
	`fechaFin` TIMESTAMP NOT NULL COMMENT 'TimeStamp con fecha y hora de fin de la practica',
	`Duracion_Horas` DECIMAL(24,4) NULL,
	`Grupos_Reservas` TEXT NULL COLLATE 'utf8mb4_general_ci',
	`Grupo1_Reservas` INT(2) NULL COMMENT 'Identificador del grupo de practicas',
	`Grupo2_Reservas` INT(2) NULL COMMENT 'Identificador del grupo de practicas',
	`idAsignatura_CV` INT(11) NULL COMMENT 'Asignatura con cortina virtual',
	`Grupo1_CV` INT(11) NULL COMMENT 'Grupo que se imparte junto',
	`Grupo2_CV` INT(11) NULL COMMENT 'Grupo que se imparte junto'
) ENGINE=MyISAM;

-- Volcando estructura para tabla tau.virtuales
CREATE TABLE IF NOT EXISTS `virtuales` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Identificador',
  `idAsignatura` int(11) DEFAULT NULL COMMENT 'Asignatura con cortina virtual',
  `grupo1` int(11) DEFAULT NULL COMMENT 'Grupo que se imparte junto',
  `grupo2` int(11) DEFAULT NULL COMMENT 'Grupo que se imparte junto',
  PRIMARY KEY (`id`),
  KEY `FK_vinculadas_asig_idx` (`idAsignatura`,`grupo1`) USING BTREE,
  CONSTRAINT `virtuales_asignaturas_idAsignatura_fk` FOREIGN KEY (`idAsignatura`) REFERENCES `asignaturas` (`idAsignatura`)
) ENGINE=InnoDB AUTO_INCREMENT=33 DEFAULT CHARSET=utf8mb4 COMMENT='Tabla que establece las vinculaciones entre asignaturas.';
```

```
-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `cortinas_reservas_sigma`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `cortinas_reservas_sigma` AS select
`r`.`idAsignatura` AS `IDASIGNATURA`,`r`.`fechaInicio` AS `fechaInicio`,(case when ((hour(`r`.`fechaInicio`) %
2) = 0) then 'Par' else 'Impar' end) AS `Hora_Par_Impar`,`r`.`fechaFin` AS `fechaFin`,
(timestampdiff(MINUTE,`r`.`fechaInicio`,`r`.`fechaFin`) / 60) AS `Duracion_Horas`,min(`r`.`idespacio`) AS
`idespacio`,min(`r`.`idgrupo`) AS `Grupo1`,max(`r`.`idgrupo`) AS `Grupo2`,group_concat(`r`.`idgrupo` order
by `r`.`idgrupo` ASC separator ',') AS `IDGRUPOS`,count(`r`.`idgrupo`) AS `Total_Grupos`,`r`.`procesado` AS
`procesado`,`r`.`asignado` AS `asignado` from `reservas` `r` where isnull(`r`.`killed`) group by
`r`.`idAsignatura`,`r`.`fechaInicio` having (count(`r`.`idgrupo`) > 1)
;

-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `resto_reservas_sigma`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `resto_reservas_sigma` AS select `r`.`id` AS
`id`,`r`.`idAsignatura` AS `IDASIGNATURA`,`r`.`idgrupo` AS `IDGRUPO`,`r`.`fechaInicio` AS `fechaInicio`,(case
when ((hour(`r`.`fechaInicio`) % 2) = 0) then 'Par' else 'Impar' end) AS `Hora_Par_Impar`,`r`.`fechaFin` AS
`fechaFin`,(timestampdiff(MINUTE,`r`.`fechaInicio`,`r`.`fechaFin`) / 60) AS `Duracion_Horas`,`r`.`idespacio`
AS `IDESPACIO`,`r`.`numpractica` AS `NUMPRACTICA`,`r`.`observaciones` AS `OBSERVACIONES`,`r`.`asignado`
AS `asignado` from ((`reservas` `r` join `asignaturas` `a` on(((`r`.`idAsignatura` = `a`.`idAsignatura`) and
(`a`.`manual` = 0) and (`a`.`activo` = 1)))) left join `conflictos_restricciones` `cr` on((`r`.`id` = `cr`.`idreserva`)))
where ((`r`.`idgrupo` not between 50 and 59) and (cast(`r`.`procesado` as date) = curdate()) and
isnull(`r`.`killed`) and isnull(`cr`.`idreserva`))
;

-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `restricciones_para_los_que_tienen_solo_dos_espacios`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW
`restricciones_para_los_que_tienen_solo_dos_espacios` AS select `restricciones`.`idAsignatura` AS
`idAsignatura`,`restricciones`.`numpractica` AS `numpractica`,min(`restricciones`.`codespacio`) AS
`opcion1`,max(`restricciones`.`codespacio`) AS `opcion2` from `restricciones` group by
`restricciones`.`idAsignatura`,`restricciones`.`numpractica` having (count(0) = 2)
;

-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `restricciones_para_los_que_tienen_solo_tres_espacios`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW
`restricciones_para_los_que_tienen_solo_tres_espacios` AS select distinct `r1`.`idAsignatura` AS
`idAsignatura`,`r1`.`numpractica` AS `numpractica`,(select `r2`.`codespacio` from `restricciones` `r2` where
((`r2`.`idAsignatura` = `r1`.`idAsignatura`) and (`r2`.`numpractica` = `r1`.`numpractica`) and (`r2`.`prioridad`
= 1))) AS `opcion1`,(select `r3`.`codespacio` from `restricciones` `r3` where ((`r3`.`idAsignatura` =
`r1`.`idAsignatura`) and (`r3`.`numpractica` = `r1`.`numpractica`) and (`r3`.`prioridad` = 2))) AS `opcion2`,
(select `r4`.`codespacio` from `restricciones` `r4` where ((`r4`.`idAsignatura` = `r1`.`idAsignatura`) and
(`r4`.`numpractica` = `r1`.`numpractica`) and (`r4`.`prioridad` = 3))) AS `opcion3` from `restricciones` `r1`
where (exists(select 1 from `restricciones` `r5` where ((`r5`.`idAsignatura` = `r1`.`idAsignatura`) and
(`r5`.`numpractica` = `r1`.`numpractica`) and (`r5`.`prioridad` = 1)) limit 1) and exists(select 1 from
`restricciones` `r6` where ((`r6`.`idAsignatura` = `r1`.`idAsignatura`) and (`r6`.`numpractica` =
`r1`.`numpractica`) and (`r6`.`prioridad` = 2)) limit 1) and exists(select 1 from `restricciones` `r7` where
((`r7`.`idAsignatura` = `r1`.`idAsignatura`) and (`r7`.`numpractica` = `r1`.`numpractica`) and (`r7`.`prioridad`
= 3)) limit 1))
;
```

```
-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `restricciones_para_los_que_tienen_solo_un_espacio`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW
`restricciones_para_los_que_tienen_solo_un_espacio` AS select `restricciones`.`idAsignatura` AS
`idAsignatura`,`restricciones`.`numpractica` AS `numpractica` from `restricciones` group by
`restricciones`.`idAsignatura`,`restricciones`.`numpractica` having (count(0) = 1)
;

-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `restricciones_reservas_de_solo_dos_espacios`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW
`restricciones_reservas_de_solo_dos_espacios` AS select distinct `r`.`idAsignatura` AS
`idAsignatura`,`a`.`manual` AS `manual`,`a`.`activo` AS `activo`,`r`.`numpractica` AS
`numpractica`,`u`.`opcion1` AS `opcion1`,`u`.`opcion2` AS `opcion2` from ((`restricciones` `r` join
`asignaturas` `a` on(((`r`.`idAsignatura` = `a`.`idAsignatura`) and (`a`.`manual` = 0) and (`a`.`activo` = 1))))
join `restricciones_para_los_que_tienen_solo_dos_espacios` `u` on(((`r`.`idAsignatura` = `u`.`idAsignatura`)
and (`r`.`numpractica` = `u`.`numpractica`)))) where `r`.`idAsignatura` in (select `reservas`.`idAsignatura`
from `reservas` where ((`reservas`.`idgrupo` not between 50 and 59) and (cast(`reservas`.`procesado` as
date) = curdate()) and isnull(`reservas`.`killed`))) order by `r`.`idAsignatura`
;

-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `restricciones_reservas_de_solo_tres_espacios`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW
`restricciones_reservas_de_solo_tres_espacios` AS select distinct `r`.`idAsignatura` AS
`idAsignatura`,`a`.`manual` AS `manual`,`a`.`activo` AS `activo`,`r`.`numpractica` AS
`numpractica`,`u`.`opcion1` AS `opcion1`,`u`.`opcion2` AS `opcion2`,`u`.`opcion3` AS `opcion3` from
((`restricciones` `r` join `asignaturas` `a` on(((`r`.`idAsignatura` = `a`.`idAsignatura`) and (`a`.`manual` = 0)
and (`a`.`activo` = 1)))) join `restricciones_para_los_que_tienen_solo_tres_espacios` `u`
on(((`r`.`idAsignatura` = `u`.`idAsignatura`) and (`r`.`numpractica` = `u`.`numpractica`)))) where
`r`.`idAsignatura` in (select `reservas`.`idAsignatura` from `reservas` where ((`reservas`.`idgrupo` not
between 50 and 59) and (cast(`reservas`.`procesado` as date) = curdate()) and isnull(`reservas`.`killed`)))
order by `r`.`idAsignatura`
;

-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `restricciones_reservas_de_un_solo_espacio`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW
`restricciones_reservas_de_un_solo_espacio` AS select `r`.`idAsignatura` AS `idAsignatura`,`a`.`manual` AS
`manual`,`a`.`activo` AS `activo`,`r`.`numpractica` AS `numpractica`,`r`.`codespacio` AS
`codespacio`,`r`.`prioridad` AS `prioridad` from ((`restricciones` `r` join `asignaturas` `a`
on(((`r`.`idAsignatura` = `a`.`idAsignatura`) and (`a`.`manual` = 0) and (`a`.`activo` = 1)))) join
`restricciones_para_los_que_tienen_solo_un_espacio` `u` on(((`r`.`idAsignatura` = `u`.`idAsignatura`) and
(`r`.`numpractica` = `u`.`numpractica`)))) where `r`.`idAsignatura` in (select distinct `reservas`.`idAsignatura`
from `reservas` where ((`reservas`.`idgrupo` not between 50 and 59) and (cast(`reservas`.`procesado` as
date) = curdate()) and isnull(`reservas`.`killed`)))
;
```

```
-- Eliminando tabla temporal y crear estructura final de VIEW
DROP TABLE IF EXISTS `Total_CV_coincidentes_Reservas`;
CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `Total_CV_coincidentes_Reservas` AS select distinct `r`.`IDASIGNATURA` AS `idAsignatura_Reservas`,`a`.`manual` AS `MANUAL`,`a`.`activo` AS `ACTIVO`,`r`.`fechaInicio` AS `fechaInicio`,`r`.`Hora_Par_Impar` AS `Hora_Par_Impar`,`r`.`fechaFin` AS `fechaFin`,`r`.`Duracion_Horas` AS `Duracion_Horas`,`r`.`IDGRUPOS` AS `Grupos_Reservas`,`r`.`Grupo1` AS `Grupo1_Reservas`,`r`.`Grupo2` AS `Grupo2_Reservas`,`v`.`idAsignatura` AS `idAsignatura_CV`,`v`.`grupo1` AS `Grupo1_CV`,`v`.`grupo2` AS `Grupo2_CV` from ((`cortinas_reservas_sigma` `r` join `asignaturas` `a` on(((`r`.`IDASIGNATURA` = `a`.`idAsignatura`) and (`a`.`manual` = 0) and (`a`.`activo` = 1)))) join `virtuales` `v` on(((`r`.`IDASIGNATURA` = `v`.`idAsignatura`) and (((`r`.`Grupo1` = `v`.`grupo1`) and (`r`.`Grupo2` = `v`.`grupo2`)) or ((`r`.`Grupo1` = `v`.`grupo2`) and (`r`.`Grupo2` = `v`.`grupo1`)))))) where ((`r`.`Grupo1` not between 50 and 59) and (`r`.`Grupo2` not between 50 and 59) and (cast(`r`.`procesado` as date) = curdate()))
;

/*!40103 SET TIME_ZONE=IFNULL(@OLD_TIME_ZONE, 'system') */;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IFNULL(@OLD_FOREIGN_KEY_CHECKS, 1) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40111 SET SQL_NOTES=IFNULL(@OLD_SQL_NOTES, 1) */;
```

PROCESO PYTHON DE CARGA DE DATOS

Se han estructurado como procesos independientes que leen un fichero de configuracion y pueden lanzarse de forma independiente.

Fichero de Configuración, contiene los datos de conexión a bbdd, los diferentes ficheros que seran procesados y la conexión al api de odile que nos devuelve los datos de asignaturas de nuestro departamento.


```
{
   "db_host": "danae03.unizar.es",
   "db_user": "tauadm",
   "db_password": "aEdi-660/pivEry",
   "db_name": "tau",
   "virtualeslsi": "D:\\Proyectos\\DatosExcel\\Cortina virtual LSI.xlsx",
   "virtualesatc": "D:\\Proyectos\\DatosExcel\\Cortina virtual ATC.xlsx",
   "virtualesisa": "D:\\Proyectos\\DatosExcel\\Cortina virtual ISA.xlsx",
   "restriccioneslsi":"D:\\Proyectos\\DatosExcel\\Restricciones laboratorios LSI.xlsx",
   "restriccionesatc":"D:\\Proyectos\\DatosExcel\\Restricciones laboratorios ATC.xlsx",
   "restriccionesisa":"D:\\Proyectos\\DatosExcel\\Restricciones laboratorios ISA.xlsx",
   "lugaresisa":"D:\\Proyectos\\DatosExcel\\Lugar practicas ISA.xlsx",
   "lugaresatc":"D:\\Proyectos\\DatosExcel\\Lugar practicas ATC.xlsx",
   "lugareslsi":"D:\\Proyectos\\DatosExcel\\Lugar practicas LSI.xlsx",
   "url": "https://docenciadiis:docenfacildiis@odile.unizar.es/api/asignaturaGrupoProfesor/
2025/110/5007",
   "reservas": "D:\\Proyectos\\DatosExcel\\reservas\\",
   "calendario": "D:\\Proyectos\\DatosExcel\\calendario.csv",
   "manuales": "D:\\Proyectos\\DatosExcel\\reservas_manual_25_26.xlsx"
}
```

TratarJsonOdile.py: Carga los datos de Asignaturas de Odile.

Lee un archivo **config.json** que contiene los parámetros de conexión a la base de datos y la URL del servicio web de Odile.
Establece una conexión con MySQL usando los datos de configuración.
Realiza una petición HTTP GET a la URL especificada en la configuración.
Convierte la respuesta JSON en una estructura de datos de Python (lista de diccionarios), verifica que el JSON recibido sea una lista y que no esté vacía y recorre cada elemento (registro) del JSON.
Para cada registro:

- Si el código de asignatura (codAsignatura) no se ha procesado antes, lo añade a un conjunto para evitar duplicados.
- Extrae los campos relevantes para la base de datos: código y nombre de la asignatura, plan, área, centro, cuatrimestre, y si está vinculada.
- Calcula un acrónimo a partir del nombre del plan y de la asignatura.
- Inserta los datos en la tabla asignaturas de la base de datos, evitando duplicados por codAsignatura.

Una vez estan dadas de alta las asignaturas en la tabla correspondiente hay que pasar a definir los campos determinados en las excel de lugares y cargar las restricciones y cortinas virtuales que nos han indicado los profesores mediante excel.

TratarExcel.py: Carga la tabla restricciones, la tabla virtuales y rellena los flag que tenemos en asignaturas.

Este script automatiza la importación y actualización de datos de virtuales, restricciones y lugares donde se va ha impartir las practicas.
Lee el archivo config.json para obtener los parámetros de conexión a la base de datos y las rutas de los archivos Excel a procesar.

- "virtualeslsi": "D:\\Proyectos\\DatosExcel\\Cortina virtual LSI.xlsx",
- "virtualesatc": "D:\\Proyectos\\DatosExcel\\Cortina virtual ATC.xlsx",
- "virtualesisa": "D:\\Proyectos\\DatosExcel\\Cortina virtual ISA.xlsx",
- "restriccioneslsi":"D:\\Proyectos\\DatosExcel\\Restricciones laboratorios LSI.xlsx",
- "restriccionesatc":"D:\\Proyectos\\DatosExcel\\Restricciones laboratorios ATC.xlsx",
- "restriccionesisa":"D:\\Proyectos\\DatosExcel\\Restricciones laboratorios ISA.xlsx",
- "lugaresisa":"D:\\Proyectos\\DatosExcel\\Lugar practicas ISA.xlsx",
- "lugaresatc":"D:\\Proyectos\\DatosExcel\\Lugar practicas ATC.xlsx",
- "lugareslsi":"D:\\Proyectos\\DatosExcel\\Lugar practicas LSI.xlsx",

Importante que ningún nombre tenga acentos o caracteres raros para evitar problemas.

Lo primero que hacemos por si hay que pasar el script varias veces en vaciar (TRUNCATE) las tablas virtuales y restricciones para evitar duplicados antes de insertar nuevos datos.

Recorre las claves del archivo de configuración y, según el nombre de la clave, realiza diferentes acciones:

a) Para las excel de cortinas virtuales

- Carga el archivo Excel correspondiente.
- Lee la hoja llamada 'Hoja 1' desde la fila 4 ya que el resto son cabeceras.
- Por cada fila válida, extrae el código de asignatura y los grupos A y B.
- Inserta estos datos en la tabla virtuales.

b) Para la excel de restricciones de laboratorios.

- Carga el archivo Excel correspondiente.
- Lee la hoja 'Hoja 1' desde la fila 4.
- Por cada fila válida, extrae el código de asignatura, el número de práctica y hasta tres opciones de espacio.
- Inserta una fila en la tabla restricciones por cada opción de espacio, asignando la prioridad (1, 2 o 3).

c) Para la excel de Lugar Practicas

- Carga el archivo Excel correspondiente.
- Lee la hoja 'Hoja 1' desde la fila 4.
- Por cada fila válida, extrae información sobre la asignatura y varias características (manual, activo, aulacentro, aulateoria, rotada, sala2, sala10).
    - Manual: identifica si las reservas las va a asignar el algoritmo o se haran de forma manual.
    - Activo: identifica si la asignatura esta activa para que el algoritmo la procese.
    - Aulacentro: indica que la asignatura se va a impartir sus practicas en aulas de Centro.
    - AulaTeoria: indica que las practicas de la asignatura se van a impartir en el mismo aula que se imparte la teoria.
    - Rotada: identifica una asignatura rotada asignando el semestre rotado.
    - Sala2: identifica asignaturas que se imparten en sala2 de centro.
    - Sala10: identifica asignaturas que se imparten en sala10 de centro.
- Normaliza los valores (por ejemplo, convierte "SI", "True", etc. a 1).
- Actualiza la tabla asignaturas con estos valores para el código de asignatura correspondiente.

TratarJsonSigma.py: procesa archivos JSON de reservas, y rellena la tabla prereservas de la bbdd, cuando ha finalizado genera un archivo Excel con las reservas de asignaturas marcadas como "manuales". Como ultimo paso proceso una excel con los dias EINA para determinar el diaEINA de cada una de las reservas.

Como el resto de procesos python carga de configuración con los parametros para la conexión de bbdd y la localizacion de los ficheros json de reservas, la localizacon del csv de calendario EINA y el lugar donde se dejara la excel de reservas manuales.

- "reservas": "D:\\Proyectos\\DatosExcel\\reservas\\",
- "manuales": "D:\\Proyectos\\DatosExcel\\reservas_manual_25_26.xlsx"

Vacía la tabla prereservas para evitar duplicados antes de insertar nuevos datos.

En el procesamiento de los ficheros JSON lo que hace es buscar todos los archivos en el directorio especificado y por cada uno comprueba que la asignatura existe en nuestra tabla asignaturas, que esta activa y no es una vinculada. Si cumple todo se procesa cada elemento del fichero.

Para procesarlo si el elemento corresponde a "Prácticas de laboratorio", inserta los datos relevantes en la tabla prereservas.

- idAsignatura
- inicio
- fin
- idgrupo
- idespacio (aula)
- diaSemana
- observaciones
- fecha de procesamiento
- año de inicio de curso.

Para la generación de la Excel con la reservas que van a ser insertadas de forma manual lo que hace es una consulta contra asignaturas y prereservas que tienen el campo manual=1. Con esa informacion se crea una excel con el nombre y ruta especificados en la configuracion.

Como ultimo paso se llama a TratarCSVCalendario.py usando subprocess.run**.**

TratarCSVCalendario.py: Rellena el campo correspondiente al diaEINA en la tabla prereservas en base al calendario proporcionado.

Este proceso lo que hace es actualizar el campo diaEina de la tabla Preresvas con la información proporcionada en la excel que nos indican en la configuración.

- "calendario": "D:\\Proyectos\\DatosExcel\\calendario.csv"

Una vez que tiene todo procesado y en la bbdd se le tiene que enviar a Carlos Gracia el fichero de reservas manuales para que lo rellene y lo procesemos ya que es el caso MAS restricitivo de espacios para practicas. Cuando nos lo hace llegar lo que tenemos que hacer en lanzar el proceso TratarManuales.py

TratarManuales.py: Procesa la excel que nos ha proporcionado Carlos Gracia.

Lee la excel en su hoja Reservas Manual y rellena el campo idespacio, numpractica y observaciones de cada reserva que coincida de la tabla prereservas.

Una vez esta relleno se ha de pasar el algoritmo que gestiona los espacios y los asigna al resto de reserva.

Para la realización de comprobaciones se ha creado el proceso **sabanas.py** que lo que hace es generar dos archivos CSV que muestran, para cada día EINA definido en un array y para cada intervalo de media hora entre las 8:00 y las 22:00, las asignaturas y grupos que tienen reservas en la base de datos. Creando un fichero antes y después de una fecha de corte definida en el programa, en este caso (1 de enero de 2026).

Los diasEina evaluados son:

- `dias_eina = ['La3', 'Ma3', 'Xa3', 'Ja3', 'Va3', 'Lb3', 'Mb3', 'Xb3', 'Jb3', 'Vb3', ]`

GenerarInformeCentro.py: Procesa los datos de las reservas para generar la excel que se enviara a Centro para que incorporen los datos en Sigma.

Los datos que extrae son:

```
SELECT a.nbAsignatura AS Nombre_Asignatura,
       a.denomplan AS Titulacion,
             a.idAsignatura AS Codigo_Asignatura,
             a.curso AS Curso,
             a.cuatrimestre AS Semestre,
             r.idgrupo AS Grupo_Practicas,
             SUBSTRING(r.diaEina, 1, 2) AS Dia_EINA,
             date_format(r.fechaInicio, '%d/%m/%Y %H:%i') AS Fecha_Inicio,
             date_format(r.fechaFin, '%d/%m/%Y %H:%i') AS Fecha_Fin,
             r.idespacio AS Espacio,
             e.edificio AS Edificio
FROM asignaturas a, reservas r, espacios e
WHERE a.idAsignatura=r.idAsignatura
AND r.idespacio=e.codEspacio
ORDER BY r.idAsignatura, r.idgrupo, r.fechaInicio;
```

Genera el fichero en la carpeta del path de la aplicación.

CRONOLOGIA DE EJECUCIÓN:

1. TratarJsonOdile una vez que se carga la tabla de asignaturas se sigue.
2. TratarExcel rellena el resto de campos de la tabla de asignaturas, la tabla virtuales y la tabla restricciones en base a las excel que tiene que darnos Carlos y almacenarlas en la ruta que indicamos en el fichero config. (los nombres deben ir sin acentos)
3. TratarJsonSigma rellena la tabla prereservas con los datos de Sigma y genera la excel de reservas Manuales, una vez generada en el path de la aplicación hay que mandarsela a Carlos y esperar a que no la devuelva rellena.
4. Solo cuando nos la devuelva ejecutamos TratarManuales, se tiene que cargar antes de pasar el algoritmo, son las reservas mas restrictivas.
5. GenerarInformeCentro se ejecuta para generar la excel con los datos a meter en Sigma, la excel se crea en el path de la aplicación.