Universidad de La Habana
Facultad de Matemática y Computación

Sistema de gestión para el funcionamiento de un departamento docente en la facultad de Matemática y Computación de la UH

Autor:
Juan Carlos Casteleiro Wong

Tutores:

Msc. Fernando Rodriguez Flores

Lic. Alain Cartaya Salabarría
Lic. Daniela González Beltrán

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Fecha

github.com/username/repo

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

Resumen en español

Abstract

Resumen en inglés

Índice general

Introducción **1**

1. Preliminares **2**
1.1. Funcionamiento de la facultad 2
1.1.1. Asignación de docencia 3
1.1.2. Tribunales de tesis 3
1.2. Descripción del software 3
1.2.1. Modelo cliente-servidor 4
1.2.2. Desarrollo del servidor 4
1.2.3. Desarrollo del cliente 5

2. Descripción de las funcionalidades que se desean **8**
2.1. Informatización de la datos 8
2.2. Asignación de docencia 8
2.3. Confección de los tribunales de tesis 9

3. Descripción del diseño de la base de datos **10**
3.1. Metodología de diseño de bases de datos 10
3.1.1. Diseño conceptual 10
3.1.2. Diseño lógico 11
3.1.3. Diseño físico 12
3.2. Correcto diseño de bases de datos relacionales 12
3.2.1. Normalización 12
3.3. Modelación de la asignación de docencia 13
3.4. Modelación de los tribunales de tesis 16

4. Descripción las herramientas implementadas **19**
4.1. Salvar y llenar datos de la BD 19
4.1.1. save database 19
4.1.2. fill database 19
4.2. Generar csv 20

4.3. Generar asignaciones de docencia 20

5. Extensibilidad **21**

Conclusiones **22**

Recomendaciones **23**

Bibliografía **24**

Índice de figuras

Ejemplos de código

Introducción

Capítulo 1

Preliminares

Este capítulo provee una breve introducción a los distintos procesos que se llevan a cabo en un departamento de la Facultad de Matemática y Computación de La Universidad de La Habana, así como una descripción de las herramientas utilizadas en la solución brindada.

1.1. Funcionamiento de la facultad

Cada año en la Facultad de Matemática y Computación de La Universidad de La Habana se realizan un conjunto de procesos de planificación y administración relacionados con los cursos escolares. La ejecución de los procesos es delegada a los distintos departamentos de la facultad y en su mayoría, si no en su totalidad, se realizan actualmente de manera manual y no informatizada, por ejemplo:

- Asignación de docencia
- Confección de los tribunales de tesis
- Asignación de cursos optativos
- Asignación de alumnos ayudantes
- Planificación de las evaluaciones del semestre

A continuación se describen los procesos de asignación de docencia y confección de tribunales de tesis que son llevados a cabo en la Facultad de Matemática y Computación de La Universidad de La Habana:

1.1.1. Asignación de docencia

El proceso de asignación de docencia en un departamento consiste en, dadas las asignaturas a impartir en un semestre y el conjunto de profesores del mismo, realizar una distribución que satisfaga los requerimientos de dichas asignaturas, contemplando la experiencia y las preferencias de los profesores.

Cada asignatura tiene sus particularidades, como la cantidad de horas a impartir en conferencias, clases prácticas, seminarios, laboratorios u otras modalidades, y la cantidad de grupos de clase que reciben dicha actividad. Estas particularidades varían en dependencia de la matrícula del curso y del plan de estudio al que la asignatura pertezca.

Por ejemplo para la asignatura Modelos de Optimización I, perteneciente al plan de estudio D, para la carrera de Ciencia de la Computación que se imparte en el tercer año, se tiene un total de horas a impartir de 64, de esas, 32 horas son de conferencia y 32 horas son de clases prácticas, además se tiene que las conferencias se imparten a un solo grupo mientras que las clases prácticas a dos.

(No se si agregar algo sobre esto) Actualmente este proceso es ejecutado por una persona 'ciclano' y debe ponerse de acuerdo con los profes, muchas veces.

1.1.2. Tribunales de tesis

Para realizar la confección de un tribunal de tesis se necesitan definir tres incógnitas fundamentales: el local, la fecha y el tribunal. El tribunal debe estar compuesto por tres profesores que asuman los roles de secretario, presidente y oponente. Los profesores que integren el tribunal no pueden ser tutores o cotutores de la tesis en cuestión, y deben tener dominio sobre el tema que será presentado. Para la elección de la fecha y el local se debe tener en cuenta que la defensa de una tesis dura aproximadamente 45 minutos.

(No se si agregar algo sobre esto) lo mismo que en el proceso anterior, mencionar las desventajas de hacer este proceso no informatizado

1.2. Descripción del software

Para la informatización de estos procesos se implementa un sistema de gestión a través de una aplicación web, utilizando un modelo cliente-servidor. Para el desarrollo en el lado del servidor se hace uso del framework Django, en particular Django Rest Framework, aprovechando la versatilidad, seguridad, escalabilidad y mantenibilidad que el framework ofrece. Mientras que en el lado del cliente se hace uso del framework Quasar, que se basa en Vue.js, creando una single-page application (SPA) o aplicación de página única. A continuación se profundiza en el modelo y herramientas utilizadas.

1.2.1. Modelo cliente-servidor

El modelo cliente-servidor es uno de los más populares en el mundo de la informática actual, es utilizado para el desarrollo de todo tipo de aplicaciones, incluso muchos de los protocolos utilizados a diario implementan este modelo, tales como File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) y Hypertext Transfer Protocol (HTTP).

El modelo cliente-servidor puede ser definido como una arquitectura de software compuesta por los proveedores de un recurso o servicio, denominados servidores, y los solicitantes del servicio, denominados clientes. En este modelo se realiza una comunicación de procesos que implica el intercambio de datos tanto por parte del cliente como del servidor, realizando cada uno funciones diferentes. La comunicación se realiza frecuentemente a través de una red informática, pero tanto el cliente como el servidor pueden residir en un mismo sistema.

Entre las principales ventajas que ofrece este modelo se encuentran:

- centralización de los recursos, los accesos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- división del procesamiento de la aplicación en varios dispositivos.
- permite la escalabilidad del sistema, en cualquier momento se puede mejorar la capacidad de procesamiento aumentando la cantidad de servidores sin que el funcionamiento de la red se vea afectado.
- la separación en cliente y servidor permite el uso de tecnologías enfocadas en las labores específicas que realiza cada uno.

1.2.2. Desarrollo del servidor

Para la implementación de los servicios que consumen los clientes del sistema de gestión se hace uso del framework Django. Django es un framework web de alto nivel, gratuito y de código abierto que permite el desarrollo rápido de sitios web seguros y mantenibles. Entre las principales características del framework se encuentran:

- Django sigue la filosofía de diseño DRY (Don't Repeat Yourself), brindando un conjunto de funcionalidades implementadas por los desarrolladores para evitar la repetición de código en procesos comunes en el desarrollo web.
- Django implementa el patrón de diseño Model-View-Template (MVT), que consta de tres componentes esenciales Modelo, Vista, Plantilla. Estas componentes son responsables de diferentes partes del código e incluso pueden utililizarse

de forma independiente. Al ser MVT una "loosely coupled architecture" o una "arquitectura débilmente acoplada", permite de manera sencilla la integración de componentes desarrolladas en paralelo.

- Django está implementado en Python, por lo que cuenta con un extenso conjunto de bibliotecas para resolver distintas tareas. Entre las bibliotecas más utilizadas resalta Django REST framework, desarrollada para la creación de interfaces de programación de aplicaciones (APIs).

- Django proporciona un Object Relational Mapper (ORM) o mapeador relacional de objetos, permitiendo la interacción con la base de datos de forma orientada a objetos. Brinda la posibilidad de crear tablas, insertar, editar, borrar y extraer datos sin escribir consultas SQL, acelerando el proceso de desarrollo web.

- Django sigue también la filosofía de "Batteries Included" o "Baterías Incluidas" proporcionando una biblioteca estándar rica y versátil con herramientas para crear sistemas complejos sin la necesidad de instalar paquetes separados. Algunas de estas "baterías" son "Django Admin", "Django ORM", "Authentication" y "HTTP".

- Django cuenta con extensa documentación, desde la documentación oficial hasta contenido en forma de artículos, tutoriales y cursos en línea.

- Django es escalable, utiliza una arquitectura de "shared-nothing" o "nada compartido", que permite agregar hardware en cualquier nivel: servidores de bases de datos, servidores de almacenamiento en caché o servidores web (REFERENCIA).

1.2.3. Desarrollo del cliente

Dentro de las tecnologías de desarrollo web más utilizadas, sin duda, JavaScript es una de las más populares en la actualidad. JavaScript es un lenguaje de programación ligero, interpretado o compilado "just-in-time" o "justo-a-tiempo" con funciones de primera clase. Es un lenguaje de programación basado en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa. Se usa principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejorar las interfaces de usarios y la creación de páginas web dinámicas[REFERENCIA WIKI].

La popularidad de JavaScript para el desarrollo web ha impulsado la creación de distintos frameworks, entre los más utilizados se encuentran:

Angular Lanzado en 2010, Google está a cargo de desarrollar y mantener Angular. La principal filosofía tras Angular es tener todo lo que el desarrollador necesita, es un framework con muchas características integradas, tales como: validaciones de formularios, envíos de solicitudes http, enrutamiento y manejo de estados, cuenta además con herramientas adicionales como: línea de comandos, interfaces para el manejo y creación de proyectos. Debido a esto muchos han llamado a Angular una plataforma, más que un framework. Google y Wix se encuentran entre las empresas más populares que utilizan Angular.

React Lanzado en 2013, Facebook desarrolla y mantiene React. A diferencia de Angular, React se enfoca en ser lo más minimalista posible, especializada en el desarrollo de interfaces de usuario, describiéndose a sí misma como una biblioteca. Cuenta con muchas menos funciones integradas que Angular por lo que se terminan utilizando muchos paquetes o dependencias de terceros al desarrollar un proyecto, ya sea para enrutamiento, manejo de estados o envío de solicitudes http. Whatsapp, Instagram, Paypal, Glassdoor y BBC son algunas de las compañías populares que usan React.

Vue.js Lanzado en 2014, su desarrollo y mantenimiento es llevado a cabo por un equipo de colaboradores a nivel mundial, siendo Evan Yu, su principal creador, un ex ingeniero de Google. Se puede decir que Vue se encuentra entre Angular y React en cuanto a la cantidad de funciones integradas que ofrece, más que React, pero menos que Angular. Sitios web como GitLab y Alibaba están usando Vue.

COMPARACION CON POPULARIDAD Y DESARROLLO
Graficos y datos sobre estos frameworks?

En los últimos años, Vue.js se ha convertido en un fuerte competidor de Angular y React. El crecimiento de Vue.js se relaciona con la simplicidad del framework, la disponibilidad de bibliotecas, componentes y materiales de aprendizaje. A pesar de que Vue.js es un framework para construir interfaces de usuario no proporciona elementos o componentes de interfaz de usuarios. Es por esto que surgen otros frameworks encima de Vue.js como una capa más de abstracción, que brindan un conjunto de componentes reutilizables y con estilo. Entre las más populares se encuentran: Vuetify, Bootstrap Vue y Quasar. Este último fue el escogido para el desarrollo del front-end del sistema de gestión.

Quasar Es un framework de código abierto desarrollado para Vue.js, que permite el desarrollo aplicaciones/sitios web de una sola página (SPA), aplicaciones/sitios web renderizados del lado del servidor (SSR), aplicaciones web progresivas (PWA),

incluso aplicaciones móviles y de escritorio, reutilizando el mismo código fuente. Entre los principales beneficios que Quasar ofrece se encuentran:

- más de 70 componentes web personalizables y de alto rendimiento para cubrir la mayoría de las necesidades de la web.
- buenas prácticas integradas tales como: compresión de código, manejo avanzado de caché, mapeo de fuentes y carga diferida.
- Quasar-CLI, una herramienta que genera de forma automática la estructura del proyecto en cuestión.
- soporte para varios idiomas, incluyendo compatibilidad para idiomas RLT (right to left), tanto para los componentes de Quasar como para el código del desarrollador.
- amplia comunidad y documentación.
- ofrece una versión UMD (Definición de módulo unificado) para la migraración de proyectos existentes sin necesidad de ningún proceso de compilación.
- promueve las buenas prácticas para mantener la seguridad de las aplicaciones desarrolladas con Quasar, en la documentación oficial se le dedica un capítulo a este tema [REF].

Capítulo 2

Descripción de las funcionalidades que se desean

El objetivo principal de este trabajo es la creación de una herramienta, sitio web o portal para informatizar y automatizar distintos procesos que se llevan a cabo en un departamento, tales como:

- asignación de docencia
- confección de tribunales de tesis
- (mencionar todos los procesos que desea informatizar y automatizar el profe Fernando?)
- automatizar los procesos anteriores

Para el desarrollo de un sistema de gestión que agrupe estos procesos se debe partir de la informatización de los datos necesarios para la ejecución de estas tareas. Por tanto se hace necesario implementar un mecanismo que permita ingresar, modificar y eliminar la información.

2.1. Informatización de la datos

2.2. Asignación de docencia

Una vez el sistema cuente con todos los datos que intervienen en la asignación de docencia, se desea implementar un mecanismo que permita la asignación de profesores a tareas o actividades relacionadas con las asignatura (conferencias, clases prácticas, laboratorios, seminarios, entre otras). Se quiere además la generación de un documento CSV que agrupe esta información.

2.3. Confección de los tribunales de tesis

Se desea implementar la funcionalidad de creación de tribunales de tesis, dada una tesis poder asignar los profesores que conformarán el tribunal en los roles de secretario, presidente y oponente, así como un lugar y fecha para la defensa de la misma. De igual forma se desea la generación de un documento CSV con esta información.

Capítulo 3

Descripción del diseño de la base de datos

El desarrollo de la informática y la computación ha permitido el almacenamiento de grandes cantidades de datos en espacios físicos limitados. Actualmente los sistemas de bases de datos juegan un papel fundamental en el desarrollo de todo tipo de sistemas computacionales.

El diseño de base de datos es un proceso fundamental a la hora de modelar el conjunto de datos y las operaciones que se deseen realizar sobre ellos. Un correcto diseño de la base de datos es esencial para garantizar la consistencia de la información, eliminar datos redundantes, ejecutar consultas de manera eficiente y mejorar el rendimiento de la base de datos [4].

3.1. Metodología de diseño de bases de datos

La complejidad de la información y la cantidad de requisitos que se deseen modelar en un sistema computacional hacen que el diseño de una base de datos no sea una tarea sencilla. Por tanto es común es común descomponer el proceso de diseño en tres etapas fundamentales: diseño conceptual, diseño lógico y diseño físico.

3.1.1. Diseño conceptual

En esta fase se obtiene como resultado una representación abstracta y de alto nivel de la realidad a partir de las especificaciones de requisitos de usuario [REF libro]. El diseño conceptual comienza con la identificación de las necesidades de los usuarios, que a menudo se pueden obtener a través de: examinar la documentación existente como formularios, observando y analizando el procesamiento de la información en el proceso que se desea modelar o mediante entrevistas a los usuarios finales [2].

El objetivo del diseño conceptual es describir el contenido de información de la base de datos, mediante un esquema conceptual de la base de datos [5], que es independiente del SGBD que se utilice para manipular la base de datos. Los programadores o diseñadores utilizan modelos conceptuales de datos para la construcción de esquemas. El modelo entidad-relación es uno de los más utilizados para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976 [REF], y está formado por un conjunto de conceptos que permiten describir la realidad mediante representaciones gráficas y lingüísticas.

La metodología utilizada para el desarrollo del modelo conceptual de datos para la modelación de los procesos de asignación de docencia y confección de tribunales de tesis fue la metodología genérica MER/XX, que se basa en un enfoque híbrido entre el modelo entidad/relacional extendido y los conceptos de la modelación orientada a objetos [4].

El modelo entidad-relación extendido describe con un alto nivel de abstracción el significado de los datos, las relaciones entre ellos y las reglas de negocio de un sistema de información. Sus dos elementos principales son las entidades y las relaciones, además existen extensiones al modelo básico como atributos de las entidades o cardinalidades de las relaciones, que aportan al modelo una mayor expresividad.

3.1.2. Diseño lógico

Es el proceso de transformar el esquema conceptual del dominio de la aplicación que se obtiene en la fase anterior, en un esquema para el modelo de datos subyacente a un SGBD particular. Existen diferentes modelos matemáticos utilizados para el diseño lógico de las bases de datos, tales como: el modelo de listas invertidas[REF], el modelo jerárquico[REF], el modelo de redes[REF] y el modelo relacional[REF].

El modelo relacional fue propuesto por Edgar Frank Codd en 1970. Es un modelo que se basa en la lógica de predicados y en la teoría de conjuntos, donde todos los datos se representan en términos de tuplas, agrupados en relaciones.

El modelo relacional fue el primer modelo de base de datos que se describió en términos matemáticos formales. A pesar de que existen implementaciones del modelo de base de datos relacional como lo definió originalmente Codd, no han tenido éxito popular hasta el momento. No obstante al modelo relacional se le atribuye un gran valor por su desarrollo teórico, que ha sido fundamento de muchos de los sistemas de gestión de bases de datos relacionales que se utilizan hoy en la actualidad, como: MySQL, Oracle, SQL Server, PostgreSQL y Microsoft Access.

El resultado de esta fase consiste en una descripción de las estructuras de datos utilizadas para almacenar la base de datos[5], que se ajuste al modelo que utilice el SGBD. Esto quiere decir que, si el modelo conceptual es expresado como un modelo entidad-relación y el modelo lógico utilizado en el SGBD es el modelo relacional,

entonces las entidades, relaciones y los atributos del modelo entidad-relación deben representarse como relaciones.

NOTA: ACLARAR el concepto de relación en el modelo relacional

3.1.3. Diseño físico

En esta etapa se transforma la estructura obtenida en la etapa del diseño lógico, con el objetivo de conseguir mayor eficiencia. Es necesario evaluar los aspectos de implementación física relacionados al SGBD que se utilice. Por ejemplo: si se trabaja con una base de datos relacional, la transformación de la estructura puede consistir en crear una nueva relación que sea la combinación de varias relaciones o separar una relación en varias relaciones o añadir algún atributo calculable a una relación.

3.2. Correcto diseño de bases de datos relacionales

El diseño de una base de datos mediante el enfoque relacional, es una tarea no determinista, es posible obtener distintos esquemas relacionales como propuestas de la base de datos. Un diseño incorrecto de una base de datos puede no responder apropiadamente a las exigencias del proceso que se modela, y puede conllevar a la generación de dificultades o errores en el acceso a los datos. Entre los principales errores o dificultades que se pueden generar se encuentran:

- **Redundancia en los datos**, provoca un aumento innecesario del tamaño de la base de datos, disminuyendo la eficiencia, además puede provocar inconsistencia de los datos que puede conducir a la corrupción de los mismos.
- **Violación de la integridad de los datos**, el término "integridad de los datos" se refiere a la correctitud y completitud de la información en una base de datos. Cuando los datos son modificados con sentencias INSERT, DELETE o UPDATE la integridad de los datos puede comprometerse.

Para lograr un correcto diseño de la base de datos se recomienda aplicar un método formal de análisis a cada uno de los esquemas obtenidos intuitivamente durante la fase de diseño conceptual, que se conoce como proceso de normalización.

3.2.1. Normalización

La normalización de una base de datos es un proceso que consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación

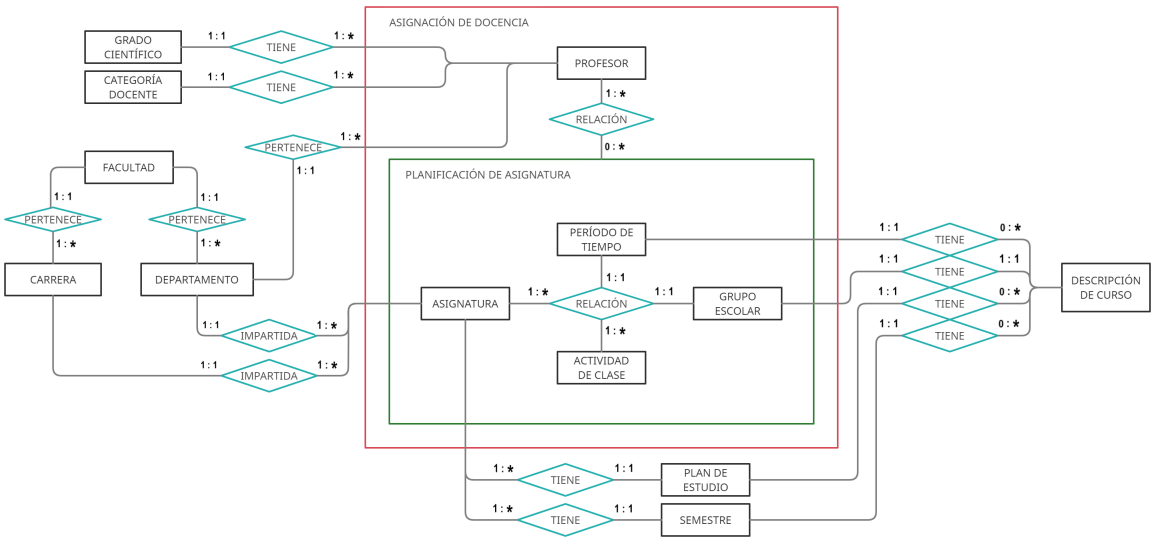
(diseño conceptual) al modelo relacional (diseño lógico), con el objetivo de minimizar la redundancia de datos. Cada regla se denomina una "forma normal", si se cumple la primera regla, se dice que la base de datos está en "primera forma normal", si se cumplen las tres primeras reglas se dice que la base de datos está en "tercera forma normal". El proceso de normalización se puede caracterizar como la transformación sucesiva de una colección de relaciones hacia una forma más restrictiva, de modo que mientrás mas profundo sea el nivel de normalización menor será la redundancia que albergue la base de datos [3]. A continuación se introducen las definiciones de las reglas formales:

- **Primera Forma Normal (1FN)**: una relación está en 1FN si y solo si, para cada ocurrencia de la relación, toda tupla contiene exactamente un valor del dominio subyacente en cada atributo.
- **Segunda Forma Normal (2FN)**: una relación está en 2FN si, además de estar en 1FN, todos los atributos que no forman parte de alguna llave candidata constituyen información acerca de la(s) llave(s) completa(s) y no de algún subconjunto de ella(s).
- **Tercera Forma Normal (3FN)**: una relación está en 3FN si, además de estar en 2FN, los atributos que no forman parte de alguna llave candidata constituyen información solo acerca de la(s) llave(s) y no acerca de otros atributos.

Aunque existen otros niveles de normalización como la "Forma Normal de Boyce-Codd (BCFN)", la "Cuarta Forma Normal(4FN)" y la "Quinta Forma Normal (5FN)", se considera que una base de datos en 3FN presenta niveles nulos o admisibles de redundancia en los datos[1].

3.3. Modelación de la asignación de docencia

En la fase de diseño conceptual se modelaron las entidades fundamentales que intervienen en el proceso de la asignación de docencia, así como las interrelaciones que se establecen entre ellas y se obtuvo el siguiente esquema a partir del modelo entidad-relación extendido.



Con el objetivo de simplificar la representación del modelo entidad-relación no se agregaron a la imagen los atributos correspondientes a cada entidad, por lo que a continuación se describen las entidades en mayor profundidad.

FACULTAD: Representa las facultades de la Universidad de La Habana. El nombre de la facultad se modela como un atributo, mientras que las carreras que se estudian en la facultad y los departamentos que pertenecen a ella, se modelan como relaciones con las entidades CARRERA y DEPARTAMENTO, respectivamente. Una facultad tiene uno o muchos departamentos, y en una facultad se pueden estudiar una o más carreras.

CARRERA: Representa las carreras que se estudian en la Universidad de La Habana. El nombre de la carrera se modela como un atributo, mientras que la facultad a la que pertenece la carrera y las asignaturas que se imparten en ella, se modelan como relaciones con las entidades FACULTAD y ASIGNATURA, respectivamente. Una carrera pertenece a una única facultad y en una carrera se imparten una o muchas asignaturas.

DEPARTAMENTO: Representa los departamentos de una facultad de la Universidad de La Habana. El nombre del departamento se modela como un atributo, mientras que los profesores que pertenecen a un departamento, las asignatura que son atendidas por el departamento y la facultad a la que pertenece el departamento, se modelan como relaciones con las entidades PROFESOR, FACULTAD y ASIGNATURA, respectivamente. Un departamento pertenece a una única facultad, un

departamento atiende una o muchas asignaturas y a un departamento pertenecen uno o muchos profesores.

PROFESOR: Agrupa los datos asociados a los profesores. Los campos nombre y apellidos se modelan como atributos, mientras que, la categoría docente, el grado científico de un profesor y el departamento al que pertenece, se modelan como relaciones con las entidades CATEGORÍA DOCENTE, GRADO CIENTÍFICO y DEPARTAMENTO, respectivamente. Un profesor pertenece a un único departamento y puede tener solo un grado científico y una categoría docente.

CATEGORÍA DOCENTE: Representa la categoría docente que tienen los profesores. El nombre de la categoría docente se modela como un atributo. Pueden existir uno o muchos profesores con una misma categoría docente.

GRADO CIENTÍFICO: Representa el grado científico que tienen los profesores. El nombre del grado científico se modela como un atributo. Pueden existir uno o muchos profesores con un mismo grado científico.

ASIGNATURA: Agrupa los datos asociados a las asignaturas. Los campos nombre de la asignatura y cantidad de horas totales a impartir, se modelan como atributos, mientras que el plan de estudio asociado a la asignatura, el semestre en el que se imparte, el departamento responsable de la asignatura y la carrera a la que pertenece, se modelan como relaciones con las entidades PLAN DE ESTUDIO, SEMESTRE, DEPARTAMENTO y CARRERA respectivamente. Una asignatura es atendida por un único departamento, pertenece a una única carrera, se imparte en un único semestre y tiene un único plan de estudio. Las asignaturas que se imparten de forma anual son representadas como dos asignaturas independientes.

PLAN DE ESTUDIO: Representa el plan de estudio por el que se rige una asignatura. El nombre del plan de estudio se modela como un atributo. Pueden existir una o muchas asignaturas con el mismo plan de estudio.

SEMESTRE: Representa los semestres asociados a una carrera. El nombre del semestre se modela como un atributo. Pueden existir una o muchas asignaturas que se imparten en el mismo semestre.

PERÍODO DE TIEMPO: Representa los períodos de tiempo del año. El nombre del período de tiempo se modela como un atributo. Pueden existir una o muchas asignaturas que se imparten en el mismo período de tiempo.

GRUPO ESCOLAR: Representa los años académicos asociados a una carrera, como por ejemplo: Matemática primer año (M1) o Computación tercer año (C3). El nombre del grupo escolar se modela como un atributo.

ACTIVIDAD DE CLASE: Representa los tipos de actividades que se imparten en una asignatura, tales como: conferencias, clases prácticas, seminarios, laboratorios, otros.

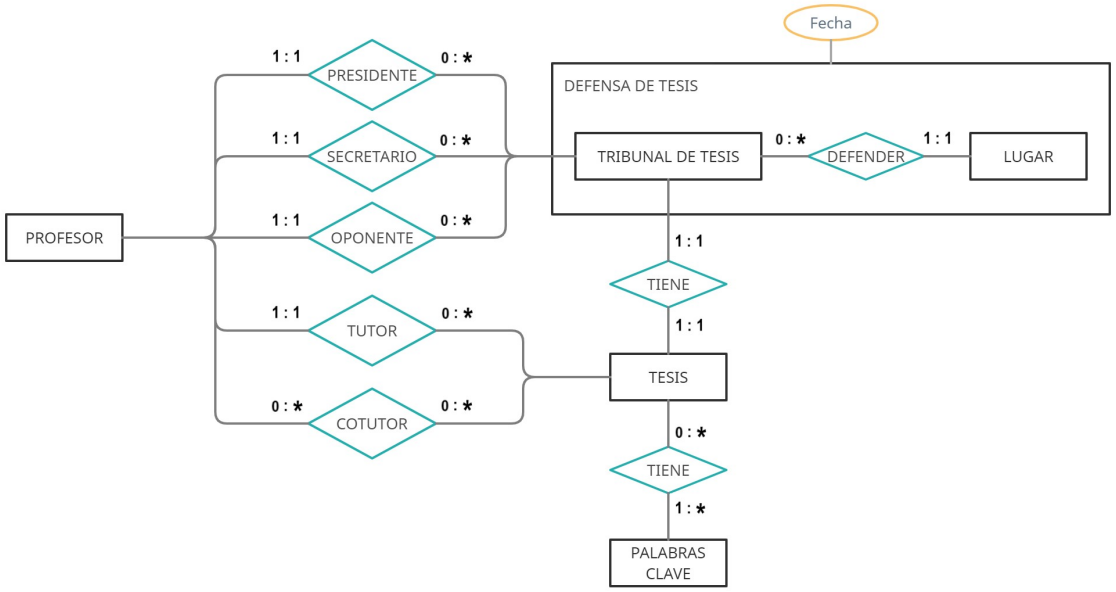
PLANIFICACIÓN DE ASIGNATURA: Se crea con el objetivo de modelar la separación de una asignatura según las actividades de clases a impartir, en un período de tiempo y para un grupo escolar específico. Está compuesta por la agregación de las entidades ASIGNATURA, ACTIVIDAD DE CLASE, GRUPO ESCOLAR y PERÍODO DE TIEMPO. Se agregan los atributos cantidad de horas para el tipo de actividad a realizar y la cantidad de grupos. Por ejemplo (poner ejemplo real de una asignatura con distintas act de clase y como queda la separación)

ASIGNACIÓN DE DOCENCIA: Representa las asignaciones de planificaciones de asignaturas a profesores. Además se agrega un atributo porciento que indica el porcentaje del total de horas a impartir que asume el profesor asignado. Un profesor puede tener asignada cero o muchas planificaciones de asignaturas y una planificación de asignatura puede estar asignada a uno o muchos profesores.

DESCRIPCIÓN DE CURSO: Representa el grupo escolar vigente en el curso actual. Está compuesta por la agregación de las entidades GRUPO ESCOLAR, PERÍODO DE TIEMPO, PLAN DE ESTUDIO y SEMESTRE. Por ejemplo, una DESCRIPCIÓN DE CURSO puede ser que el grupo escolar C4 (Computación de cuarto año), con plan de estudio E, se encuentra en el semestre 8, en el período de tiempo septiembre-diciembre.

3.4. Modelación de los tribunales de tesis

Para la modelación del proceso de confección de los tribunales de tesis, se crea un esquema basado en el modelo entidad-relacional extendido como en el proceso anterior.



TESIS: Agrupa los datos asociados a una tesis. El título de la tesis y el autor se modelan como atributos mientras que el tutor, los cotutores y las palabras claves se modelan con relaciones con las entidades PROFESOR (tutor y cotutores) y PALABRAS CLAVES. Una tesis tiene un único tutor, puede tener cero o muchos cotutores y tiene una o muchas palabras claves.

PALABRAS CLAVES:

TRIBUNAL DE TESIS:

DEFENSA DE TESIS:

LUGAR:

PROFESOR: Se utiliza la misma entidad creada en el proceso de asignación de docencia.

Thesis: Agrupa los datos asociados a una tesis, tales son: título, estudiante, tutor, posibles cotutores y palabras claves.

ThesisCommittee: Entidad que representa un tribunal de tesis, cuenta con una 'Thesis', una fecha, un lugar y tres relaciones con la entidad 'Professor' para los roles de secretario, presidente y oponente.

Los campos lugar y palabras claves fueron modelados como nomencladores en las entidades 'Place' y 'Keyword'.

Capítulo 4

Descripción las herramientas implementadas

4.1. Salvar y llenar datos de la BD

En el servidor se implementó un módulo para el trabajo con la base de datos que permite salvar el estado de la base de datos en documentos csv, de igual forma permite llenar la base de datos a partir de documentos csv.

Se crearon dos comandos para ejecutar en la terminal con el fin de realizar las tareas mencionadas previamente:

4.1.1. save database

El comando save database permite salvar tanto una tabla como la base de datos completa.

Para salvar solo una tabla se hace uso del argumento -m y luego el nombre de la tabla a salvar:

```
python manage.py save_database -m Professors
```

Para salvar la base de datos completa ejecutar solo el comando:

```
python manage.py save_database
```

4.1.2. fill database

El comando fill database permite llenar desde documentos csv tanto una tabla como la base de datos completa.

Análogamente al comando anterior para llenar solo una tabla se hace uso del argumento -m y luego el nombre de la tabla a llenar:

```
python manage.py fill_database -m Subjects
```

Para llenar la base de datos completa ejecutar solo el comando:

```
python manage.py fill_database
```

Los posibles nombres de entidades a utilizar con el parámetro -m tanto para los comandos save database como fill database son:

ClassTypes, Faculties, ScientificDegrees, TeachingCategories, Semesters, TeachingGroups, TimePeriods, Careers, StudyPlans, CarmenTable, Departments, Subjects, Professors, SubjectDescriptions, TeachingAssignments, Places, Keywords, Thesis, ThesisCommittee

4.2. Generar csv

Se implementó la funcionalidad de descargar las asignaciones de docencia y los tribunales de tesis en ficheros csv.

4.3. Generar asignaciones de docencia

Se implementó un modelo de optimización para la generación de asignaciones de docencia, pendiente computar el peso de las asignaciones (dado las preferencias de los profesores y sus habilidades)

Capítulo 5

Extensibilidad

Conclusiones

Conclusiones

Recomendaciones

Recomendaciones

Bibliografía

[1] `https://learn.microsoft.com/es-es/office/troubleshoot/access/database-normalization-description`. Accessed: 2022-07-28 (vid. pág. 13).

[2] J. Choobineh, M.V. Mannino y V.P. Tseng. «The role of form analysis in computer-aided software engineering». En: *Conceptual Modeling, Databases, and Case*. Wiley, 1992, págs. 433-445. ISBN: 0471554626 (vid. pág. 10).

[3] Lucina García Hernández y Martha Montes de Oca Richardson. «DISEÑO DE BASES DE DATOS RELACIONALES». En: *SISTEMAS DE BASES DE DATOS: Modelación y Diseño Primera Parte*. Editorial Félix Varela, 2005, págs. 111-134. ISBN: 959-258-881-3 (vid. pág. 13).

[4] Lucina García Hernández y Martha Montes de Oca Richardson. «DISEÑO DE LA BASE DE DATOS». En: *SISTEMAS DE BASES DE DATOS: Modelación y Diseño Primera Parte*. Editorial Félix Varela, 2005, págs. 40-64. ISBN: 959-258-881-3 (vid. págs. 10, 11).

[5] Lucina García Hernández y Martha Montes de Oca Richardson. «MODELOS MATEMÁTICOS DE DATOS». En: *SISTEMAS DE BASES DE DATOS: Modelación y Diseño Primera Parte*. Editorial Félix Varela, 2005, págs. 65-110. ISBN: 959-258-881-3 (vid. pág. 11).