TEC | Tecnológico de Costa Rica

LabTEC

Documentación

Ingeniería en Computadores

Bases de Datos (CE3101)

Grupo 09

Profesor:
Marco Rivera Meneses

Estudiante:
Andrés Molina Redondo | 2020129522
Ignacio Lorenzo Martínez | 2019068171
Luis Alfredo González Sánchez | 2021024482
Fabián Castillo Cerdas | 2020202938

Semestre I, 2024

Índice

Modelo conceptual (Notación de Chen)

https://lucid.app/lucidchart/3c4ed5b3-82fa-4f33-92fa-c766b50f93af/edit?viewport_loc=0%2C-407%2C1671%2C1963%2C0_0&invitationId=inv_d66ed403-21e6-49cd-8d63-cb0722e3a41e

Modelo Relacional

https://lucid.app/lucidchart/14e27f83-a9ce-44af-bd55-5562612e42c7/edit?viewport_loc=-20%2C-454%2C2744%2C3223%2C0_0&invitationId=inv_933efb44-7cfc-4c92-93f1-2cb013bfbea5

Descripción del uso de las estructuras de datos desarrolladas (Tablas)

[Base de Datos - Documentos de Google]

Análisis de Relaciones

Nombres y Tipos de Relaciones en el Modelo Relacional

Gestiona_Laboratorios (Administrador - Laboratorios)

Tipo: 1:N
Correctamente define que un administrador gestiona múltiples laboratorios, pero cada laboratorio es manejado por un solo administrador.

Gestiona_Activos (Administrador - Activos)

Tipo: 1:N
Correctamente define que un administrador gestiona múltiples activos.

Realiza_Préstamo (Operador - Préstamos)

Tipo: 1:N
Describe que un operador puede realizar varios préstamos.

Realiza_Devolución (Operador - Devoluciones)

Tipo: 1:N
Muestra que un operador puede manejar varias devoluciones.

Aprobar_Préstamo (Profesor - Préstamos)

Tipo: 1:N
Describe que un profesor puede aprobar múltiples préstamos.

Presta_Activo (Préstamos - Activos)

Tipo: N:1
Esta relación refleja que varios préstamos pueden estar vinculados a un mismo activo.

Registro_Devolución (Préstamos - Devoluciones)

Tipo: 1:1
Correctamente asocia cada préstamo con una única devolución.

Reporta_Avería (Devoluciones - Averías)

Tipo: 1:N
Muestra que una devolución puede tener asociadas múltiples averías.

Reporta_Horario (Laboratorios- HorariosLaboratorio)

Tipo: 1:N
Asocia cada laboratorio con 1 horario o con varios horarios

Solicitar_Laboratorio (Profesor - HorariosLaboratorio)

Tipo: 1:N
Asocia un profesor con 1 horario o varios horarios de los laboratorios

Descripción de Atributos y Tipos de Datos

Para facilitar la creación del script, primero detallamos los atributos y tipos de datos para cada tabla:

1. Administradores

- Correo (PK): ***VARCHAR(255)***, correo electrónico del administrador, único y no nulo debido a la restricción de clave primaria.
- Nombre: ***VARCHAR(255)***, nombre completo del administrador.
- Contraseña: ***VARCHAR(255)***, contraseña del administrador.

2. Operadores

- Carnet (PK): ***VARCHAR(100)***, identificador único del operador.
- Nombre: ***VARCHAR(255)***, nombre del operador.
- Apellido: ***VARCHAR(255)***, apellido del operador.
- FechaNacimiento: ***DATE***, fecha de nacimiento del operador.
- Correo: ***VARCHAR(255***), correo electrónico del operador, asegurado como único y no nulo.
- Contraseña: ***VARCHAR(255)***, contraseña cifrada del operador.
- Cedula: ***VARCHAR(255)***, Cedula del operador.
- Edad: ***INT***, Edad del operador.
- Aprobado: ***Boolean***, Si es True la cuenta ya fue aprobada y False si la cuenta no ha sido aprobada por un Admin.

3. SesionesOperador

- Carnet: ***VARCHAR(100)***, referencia al carnet del operador, que debe coincidir con un valor existente en la tabla Operadores (clave foránea) y no puede ser nulo.
- Fecha: ***DATE***, fecha de la sesión. Solo almacena la fecha, sin información de hora.
- HoraInicio: ***TIME***, hora exacta en que la sesión del operador comienza.
- HoraFin: ***TIME***, hora exacta en que la sesión del operador finaliza.

4. Profesores

- Cédula (PK): ***VARCHAR(100)***, número de identificación del profesor.
- Nombre: ***VARCHAR(255)***, nombre del profesor.
- Apellido: ***VARCHAR(255)***, apellido del profesor.
- Edad: ***INT***, edad del profesor.
- FechaNacimiento: ***DATE***, fecha de nacimiento del profesor.
- Correo: ***VARCHAR(255)***, correo electrónico del profesor, asegurado como único y no nulo.
- Contraseña: ***VARCHAR(255)***, contraseña cifrada del profesor.

5. Laboratorios

- Nombre (PK): ***VARCHAR(255)***, nombre único del laboratorio.
- Capacidad: ***INT***, capacidad máxima de personas.
- Computadoras: ***INT***, número de computadoras disponibles.
- Facilidades: ***VARCHAR(255***), descripción de las facilidades disponibles en el laboratorio.

6. HorariosLaboratorios

- HorarioID (PK): ***INT***, identificador único autoincremental del registro de horario para un laboratorio.
- LaboratorioNombre: ***VARCHAR(255)***, nombre del laboratorio que referencia a la tabla Laboratorios (clave foránea), asegurado como no nulo.
- Fecha: ***DATE***, fecha específica para la que se reserva el horario.
- HoraInicio: ***TIME***, hora de inicio para la reserva del horario del laboratorio.
- HoraFin: ***TIME***, hora en que termina la reserva del horario del laboratorio.

7. Activos

- Placa (PK): ***VARCHAR(100)***, identificación única del activo.
- Tipo: ***VARCHAR(100)***, tipo de activo.
- Marca: ***VARCHAR(100)***, marca del activo.
- FechaCompra: ***DATE***, fecha en que fue comprado el activo.
- Ocupado: ***Boolean***,True si el activo está prestado actualmente, False si está disponible.

- Cedula: ***VARCHAR(100)***, Cedula de profesor que tiene que hacer la aprobación.

8. Préstamos

- PréstamoID (PK): ***INT***, identificador único del préstamo.
- Placa: ***VARCHAR(100)***, identificación del activo asociado (clave foránea), asegurado como no nulo.
- Carnet: ***VARCHAR(100)***, carnet del operador que realiza el préstamo (clave foránea), asegurado como no nulo.
- FechaPréstamo: ***DATE***, fecha de realización del préstamo.
- HoraPréstamo: ***TIME***, hora del préstamo.
- EstadoDelPréstamo: ***VARCHAR(100)***, estado del préstamo (pendiente, aprobado, etc.).
- Cédula: ***VARCHAR***(100), cédula del profesor que aprueba el préstamo (clave foránea).
- NecesitaAprobacion: ***Boolean***,True si el activo necesita aprobación, False si no necesita aprobación.
- EstadoAprobacion: ***Boolean***,True si ya fue aprobado, False si aún no está aprobado.
- Entregado: ***Boolean***,True si el activo se entregó, False si aún no se entrega
- Nombre: ***VARCHAR***(100), Nombre asociado.
- Apellidos: ***VARCHAR***(100),Los apellidos asociados.
- Correo: ***VARCHAR***(100), El correo asociado.

9. Devoluciones

- DevoluciónID (PK): ***INT***, identificador único de la devolución.
- PréstamoID: ***INT***, identificador del préstamo asociado (clave foránea), asegurado como no nulo.
- Carnet: ***VARCHAR(100)***, carnet del operador que maneja la devolución (clave foránea), asegurado como no nulo.
- FechaDevolución: ***DATE***, fecha de la devolución.
- HoraDevolución: ***TIME***, hora de la devolución.

- EstadoFinalDelActivo: ***VARCHAR(255)***, estado final del activo (bueno, dañado, etc.).

10. Averías

- AveríaID (PK): ***INT***, identificador único de la avería.
- DevoluciónID: ***INT***, identificador de la devolución asociada (clave foránea), asegurado como no nulo.
- Descripción: ***VARCHAR(255)***, descripción de la avería.
- FechaDeRegistro: ***DATE***, fecha en que se registró la avería.

Resumen de Restricciones de Clave Foránea

1.SesionesOperador

FK_Carnet_Operadores: Asegura que el "**Carnet**" en "**SesionesOperador**" exista en "**Operadores**".

2.HorariosLaboratorios

FK_HorariosLaboratorios_Laboratorios: Asegura que el "**LaboratorioNombre**" en "**HorariosLaboratorios**" exista en "**Laboratorios**".

FK_CédulaProfesor: Asegura que "**CédulaProfesor**" en "**HorariosLaboratorios**" exista en "**Profesores**".

3.Prestamos

FK_Placa_Activos: Asegura que la "**Placa**" en "**Prestamos**" exista en "**Activos**".

FK_Carnet_Operadores: Asegura que el "**Carnet**" en **"Prestamos**" exista en "**Operadores**".

FK_Cedula_Profesores: Asegura que la "**Cedula**" en "**Prestamos**" exista en "**Profesores**".

4.Devoluciones

FK_Prestamo_ID: Asegura que "**PrestamoID**" en "**Devoluciones**" exista en "**Prestamos**".

FK_Carnet_Operadores: Asegura que el "**Carnet**" en "**Devoluciones**" exista en "**Operadores**".

5.Averias

FK_Devolucion_ID: Asegura que "**DevolucionID**" en "**Averias**" exista en "**Devoluciones**".

Descripción detallada de la arquitectura desarrollada

Descripción de la arquitectura:

Para el FrontEnd se utilizó Angular y bootstrap, en el backend se realizó un API REST en c#
Además este backend está conectado con una base de datos en SQL SERVER 2022

Flujos de datos y control:

El flujo de datos se utilizó en la mayoría de los casos a través de JSON en el caso entre Frontend y Backend ya que el API y sus request están hechos de manera que reciban JSON con los datos necesarios, la mayoría de estos datos se reciben en String,Int o Boolean, incluso para las fechas y horas se reciben en formato string y en el backend se realiza un parseo, esto ya que para la conexión entre Backend y base de datos SQL se necesita que los tipos concuerden y algunos tipos de Date y TimeZone no concordaban por pequeños detalles lo cual ocasiona error, por lo cual se realiza el parseo dentro del backend para que concuerde con el tipo de la base de datos
Diagramas que muestran cómo fluyen los datos a través del sistema y cómo se controlan las operaciones.

Descripción detallada de los algoritmos desarrollados

En el API se realizan diferentes Request, las cuales serían GET,POST,PUT y DELETE

En estos Request la base es la mayoría en cada uno de sus tipos, lo que cambia son los modelos que se utilizan, o los datos que son seleccionados en el query en el BACKEND para obtener los datos y conectarse a la tabla en la Base de Datos correspondiente, pero la funcionalidad es la misma para la mayoría:

- ***GET:*** En este caso hay 2 maneras de hacerlo ya sea un obtención de todos los datos de las tablas, o buscar una dupla con un identificador específico, en la mayoría de los casos sería añadiendolo después de la ruta del GET normal y utilizándolo en el backend como {id} donde id es identificador dependiendo del modelo que se utilice.
- ***POST:*** En este caso se utilizan dos modelos el modelo normal tiene por default el nombre del controlador y el segundo modelo es el nombre además de Dto, este segundo modelo se utiliza para poder recibir los datos en forma de string para que no haya problemas en el caso de de las fechas y horario,

estas se parsean a Date y Datetime para guardarse a la base de datos correctamente.

- ***PUT:*** en el caso de put pasa algo parecido al POST se utiliza los modelos Dto para evitar problemas con las fechas y horas, y estas se parsean en el BACKEND, la diferencia es que puede recibir solo un dato o varios dependiendo de la cantidad de datos que se quieran editar, además de esto se hace la ruta con un controller/{id} donde id es el identificador dependiendo de lo que se quiera editar y del controlador correspondiente, los datos que se quieren editar se envían en formato JSON.
- ***Delete:*** Este se encarga de realizar el borrado de la dupla donde se encuentre el identificar, igual que PUT se hace la petición a un api/controller/{id} donde id es el identificador, en este caso en vez de editar los datos específicos que se envíen en forma JSON, si no que en vez de esto buscará el identificador de la ruta y borrara la dupla donde coincida ese ID, este request no funciona correctamente debido a que los identificadores suelen ser PK las cuales se utilizan como FK por lo que por las restricciones de la base de datos, no se puede cambiar, y una manera de arreglar esto sería volviendolos nulls para no borrar los datos, pero la base de datos se hizo de manera que todas las FK no permitieran nulls por lo cuál debido a la seguridad de la base de datos el Request DELETE actualmente no es posible.

Plan de trabajo (Actividades realizadas por estudiante)

Problemas encontrados y soluciones

Hay diferentes problemas entre el Frontend y BackEnd que no se pudieron solucionar, en la mayoría de casos se sospecha que sea debido a la toma de datos en FrontEnd y al envío de estos mismos al Backend ya que al parecer da ciertos

errores o suelen dar errores relacionadas con las FK de la base de datos por la cual no se pueden enviar datos que no existen tablas

Conclusiones del proyecto

Concluimos que el proyecto necesita mejor comunicación en el equipo debido a que hubo varios problemas y la mayor parte de problemas fue debido a la conexión y traspaso de datos entre Frontend y Backend, también entre el backend y base de datos sql debido a que por problemas de seguridad no se conectaba hasta que se encontró la manera,además debido a la información que no poseíamos habían cosas como la migración de la base de datos al Backend que no conocíamos entonces debido a esto los datos no eran correctos y daban muchos problemas.
Se concluyo que para un proyecto de este estilo se necesita una mejor comunicación y mejor estandares a cuando paleta de colores, diseño y comentarios y estándares en el código debido a que cuando un compañero revisaba un código a veces no era claro la funcionalidad y terminaba en más errores

Recomendaciones del proyecto

Dedicarle más tiempo en grupo para declarar la manera con la que se va a manejar los datos para no tener errores en el futuro al conectar varios componentes de Angular o el FrontEnd con el Backend, además de esto es importante hablar si se utiliza una paleta de colores para el diseño, el estándar de los componentes, funcionalidades, comentarios y códigos para que sea más sencillo de entender el código entre compañeros