UNEMI
UNIVERSIDAD ESTATAL DE MILAGRO

FACULTAD DE CIENCIAS E INGENIERÍA
CARRERA DE INGENIERÍA DE SOFTWARE

TEMA:

Trabajo de investigación

AUTORES:

Matute Guamán Edison Eduardo

Avila Sanchez Adrian Alexander

Lopez Pineda Angel Alfonso

Taday Malan Juan

ASIGNATURA:

Sistemas Operativos

DOCENTE:

Javier Bermeo

FECHA DE ENTREGA:

Viernes 13/06/2025

PERIODO:

Abril 2024 a Agosto 2025

MILAGRO-ECUADOR

Contenido

Introducción

En el contexto actual de la educación, las instituciones de educación superior enfrentan un desafío constante para optimizar los procesos académicos a través de instrumentos tecnológicos. Este documento ha desarrollado una aplicación para la aplicación de gestión académica de la universidad como parte de la arquitectura de software y el dispositivo de investigación práctica de diseñador dedicado a un diseño de software avanzado con Genexus.

La acción consiste en la automatización de la carrera, el tema, el maestro y el plan de examen y la distribución de la distribución, teniendo en cuenta la validación, los subtipos, el segundo nivel en las transacciones y el uso de fórmulas. En este sentido, Genexus se utiliza como plataforma de desarrollo, teniendo en cuenta su enfoque para el modelado de conocimiento y la capacidad de generar aplicaciones completas de manera efectiva.

Las transacciones relacionadas se desarrollaron para administrar información académica. Se utilizaron reglas comerciales significativas, como la aprobación de los maestros, los pasivos en ciertas áreas, advertencias personalizadas y cálculos automáticos. Del mismo modo, las decisiones tomadas en el modelado están técnicamente justificadas y la simulación de datos se incluye en un entorno funcional.

Este trabajo demuestra no solo el área técnica del instrumento, sino también la comprensión del diseño lógico necesario para crear soluciones reales y personalizables en un sistema académico automatizado.

Objetivo

Objetivos generales

Desarrolle una aplicación funcional para la gestión académica de una institución de educación superior utilizando la herramienta Genexus como un entorno de desarrollo para automatizar procesos importantes, como el registro profesional, los temas, los maestros y la programación del plan de examen. El propósito de este desarrollo es utilizar el concepto de modelado de transacciones mejorado, la integridad de la referencia, la lógica del entorno de datos y el negocio de acuerdo con los principios de diseño de software estructurados y efectivos.

Objetivos específicos

- Analice los requisitos del sistema académico de la Universidad para identificar unidades, condiciones y reglas comerciales necesarias en el desarrollo de transacciones en Genexus.
- Transacciones modelo con relaciones complejas, incluidas las estructuras N: M, subtipos para distinguir entre roles específicos y otros niveles de elementos integrados como maestros en los exámenes.

- Implemente la integridad de las validaciones funcionales que proporcionan la integridad de los datos ingresados, como los campos obligatorios (por ejemplo, evitando la duplicación de los maestros en un solo rol) y advertencias para el usuario.
- Diseñe fórmulas automáticas que permitan cálculos de tiempo real en el sistema, como el número de maestros asignados al examen para mejorar el control y el monitoreo administrativo.
- Documente y justifique técnicamente cada decisión de modelado que explique las condiciones especificadas, el uso de subtipos y beneficios del enfoque adoptado en términos de escalabilidad y mantenimiento del sistema.
- Imitando datos reales en el entorno de prueba generado por Genexus, asegúrese de que la capacidad y validación de la validación del sistema sean cumplidas por todos los criterios funcionales y de diseño propuestos en la declaración del examen.

Marco Teórico

El diseño y desarrollo de sistemas de información académica debe utilizar conceptos técnicos clave de software, modelado de datos y herramientas de desarrollo rápido como Genexus. Los conceptos básicos teóricos que apoyan la estructura de este sistema se presentan a continuación:

1. Diseño de software orientado a datos

El diseño central de software consiste en identificar dispositivos, condiciones y limitaciones que modelan fielmente la realidad de dominio. En el contexto del sistema académico, dispositivos como la carrera, las materias y los maestros deben organizar estructuras lógicas y normalizadas que garanticen la integridad y la eficiencia de la gestión de la información. El uso de circunstancias como una de una afica (1: N) o muchas (N: M) permite representar escenarios académicos ordinarios, como temas para una carrera o maestros sobre temas.

2. Genexus como herramienta de desarrollo

Genexus es una plataforma de desarrollo basada en el modelado de conocimiento que puede generar automáticamente aplicaciones web, móviles y de escritorio. Su enfoque basado en la transacción permite que los datos apliquen reglas comerciales, generen interfaces y administren la lógica del sistema sin programación de bajo nivel. Además, Genexus admite subtipos, otros niveles de datos, validaciones automáticas y fórmulas, promoviendo el desarrollo de soluciones duraderas flexibles.

3. Subtipos y otros niveles en transacciones

El uso de subtipos en Genexus le permite definir atributos resultantes de una unidad base que es útil si tiene que distinguir entre roles o funciones sin una estructura. En este proyecto, los profesores regulares y alternativos son subtipos profesores, lo que permite que la integridad de la tabla básica se mantenga sin despido. El segundo nivel de la transacción es la propiedad que permite que la información relacionada se procese como una lista de elementos relacionados con la unidad principal. Por ejemplo, el segundo nivel se usa para dar más maestros al mismo evento académico.

4. Validación y términos comerciales

La validación es esencial para garantizar que los datos ingresados cumplan con las reglas del sistema. Estos incluyen restricciones tales como campos obligatorios (como nombre, fecha, hora), advertencias personalizadas (como teléfono de vacío) y validación lógica (por ejemplo, que el mismo maestro no es titular y reemplaza el mismo tema). Estas reglas le permiten reducir los errores, asegurar contextos del sistema y mejorar la experiencia del usuario.

5. Fórmulas automáticas y cálculos derivados

Las fórmulas de Genexus le permiten calcular los valores que provienen automáticamente de otros datos del sistema. En el caso del sistema académico desarrollado, se introduce una fórmula para contar el número de maestros asignados al examen. Esta funcionalidad mejora la visibilidad administrativa y permite confirmar condiciones como la distribución mínima necesaria de los maestros.

6. Datos y la integridad de la escalabilidad

La introducción adecuada de relaciones primarias y extranjeras, subtipos y reglas comerciales garantiza la integridad de referencia del sistema. Esto asegura que los datos siempre sean consistentes y evite las desviaciones en las operaciones como inserciones, actualizaciones o eliminación. Además, el modelo introducido está listo para escalar el futuro, lo que le permite incluir nuevas características sin poner en peligro la estructura básica del sistema.

Análisis de requerimientos

Requerimientos Funcionales

- **FNC-001:** El sistema debe contar con la funcionalidad para registrar distintas carreras, asegurando que cada una tenga un nombre único sin repeticiones.
- **FNC-002:** Debe ser posible ingresar nuevas asignaturas, garantizando que no se repitan nombres ya registrados.
- **FNC003:** Los profesores registrados en el sistema deben incluir un correo electrónico y número de teléfono válidos como parte obligatoria de su información.
- **FNC004:** La plataforma debe permitir asociar varias asignaturas a una misma carrera, admitiendo una relación múltiple entre ambas entidades.
- **FNC005:** Para cada asignatura, el sistema debe permitir la asignación de un docente principal (titular) y un docente de respaldo (suplente).
- **FNC006:** Es necesario verificar que el docente titular y el suplente asignados a una misma materia no sean la misma persona.
- **FNC007:** La herramienta debe facilitar la creación y programación de horarios de examen para las distintas asignaturas.
- **FNC008:** Al momento de asignar profesores a los exámenes, el sistema debe calcular de forma automática el número total de docentes involucrados.

Requerimientos No Funcionales

- **REQ-NF001:** La interfaz de usuario debe estar diseñada de forma clara y comprensible, facilitando la navegación y el uso sin requerir conocimientos técnicos avanzados.

- **REQ-NF002:** Es imprescindible que todos los campos obligatorios sean validados antes de guardar o procesar información.
- **REQ-NF003:** Los mensajes de advertencia o error que aparezcan durante el uso del sistema deben ser precisos y fáciles de entender, indicando claramente el motivo del problema.
- **REQ-NF004:** El sistema debe contar con mecanismos que eviten el ingreso duplicado de datos, garantizando que no se repita información previamente registrada.

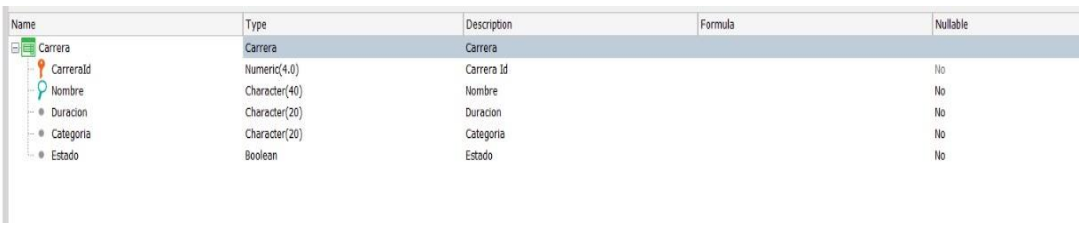
Modelado de Transacciones

Transacciones principales

Se crearon tres transacciones esenciales para el desarrollo del sistema, organizadas por las unidades más importantes en el campo académico: carrera, Asignatura y profesor . Estas transacciones son las estructuras básicas más apropiadas a las que se construirán condiciones más complejas.

Transacción Carrera

La transacción “Carrera” esta definida con atributos que nos permitirán registrar de una manera mas completa los datos de la carrera, Identificando cada carrera con su id respectivo que nos permitirá registrar e identificar cada carrera, Se incluye el atributo “Nombre” que es registrado como (Carácter) y con su respectiva validación al igual que los otros atributos es obligatorio llenar ese campo para guardar la informacion en la base de datos, el mismo proceso ocurre con la duracion de carrera que cuenta con el atributo “Duracion” que señalara los semestres que tiene dicha carrera y definiendo si es ingeniería o licenciatura con el atributo “Categoria”.



Name	Type	Description	Formula	Nullable
Carrera	Carrera	Carrera		
CarreraId	Numeric(4.0)	Carrera Id		No
Nombre	Character(40)	Nombre		No
Duracion	Character(20)	Duracion		No
Categoria	Character(20)	Categoria		No
Estado	Boolean	Estado		No

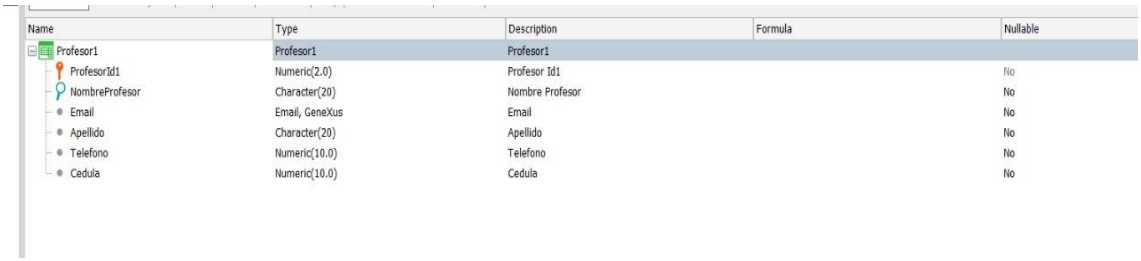
Transacción Asignatura

La transacción Asignatura representa las materia que tiene cada semestre, para diferenciarla tendrá una clave de (Id_asignatura) como clave primaria de esta transacción que transforma a cada asignatura como única con su id respectivo. El atributo principal es el nombre de la asignatura(AsignaturaNombre) que esta definida como carácter(Character) que dará información a que materia en especifico se refiere.

Name	Type	Description	Formula	Nullable
Asignatura	Asignatura	Asignatura		
Id_Asignatura	Numeric(4.0)	Id_Asignatura		No
AsignaturaNombre	Character(20)	Asignatura Nombre		No

Transacción Profesor

La transacción Profesor permite conocer y almacenar los datos de un docente de una manera más detallada, la clave primaria de la transacción es (ProfesorId1) que seria única y un identificador para relaciones posteriores, se colocaron los campos de (NombreProfesor, Apellido) que obtiene el tipo carácter(Character) y con configuraciones específicas para un excelente control de ingreso de datos en el almacenamiento, y los atributos (Email, Telefono, cedula) cada uno con sus respectivos tipos al que pertenecen que contienen la validaciones correspondientes, estos datos nos permitirán conocer la información esencial del docente.



Name	Type	Description	Formula	Nullable
Profesor1	Profesor1	Profesor1		
Profesor1d1	Numeric(2.0)	Profesor 1d1		No
NombreProfesor	Character(20)	Nombre Profesor		No
Email	Email, GeneXus	Email		No
Apellido	Character(20)	Apellido		No
Telefono	Numeric(10.0)	Telefono		No
Cedula	Numeric(10.0)	Cedula		No

Transacciones Relacionales

Transacción Asignatura Carrera

En la transacción de asignatura profesor se incorporo atributos como el número de créditos que contiene la carrera(Creditos), En información de la transacción los datos necesario que ya estaban previamente detallado en que se los llamo mediante su clave primaria, esto nos ayudo para la asignación a de carrera a cada asignatura y al mismo tiempo también obtuvimos una relación de mucho a muchos entre asignatura (Id_Asignatura) y carrera(CarreraId) utilizada como clave foránea, la información que se guarde tendrá como identificador la (AsignaturaCarreraId) que es la clave primaria de la transacción.

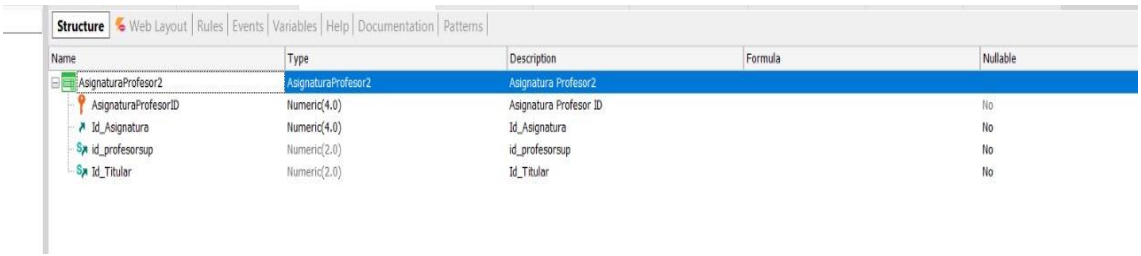
Aunque se puede identificar como clave compuesta (Id_Asignatura + CarreraId), se opto por un identificador único (AsignaturaCarreraId) por razones practicas.

- Simplifica referencias desde otras transacciones o procesos
- Mejora el rendimiento en consultas y relaciones
- Facilita la lectura y mantenimiento del modelo de datos

Transacción AsignaturaProfesor

En el desarrollo de la transacción AsignaturaProfesor se buscó representar la relación entre las asignaturas y los profesores que tienen a cargo la materia. Se diferenciaron dos roles: el profesor titular (Id_Titular) y el profesor suplente (Id_ProfesorSup). Para poder colocar estos dos roles en la misma transacción, se usaron subtipos del atributo ProfesorId, lo que permite usar la misma entidad Profesor pero en funciones distintas.

Esto se hizo con el fin de mantener la integridad referencial y también para poder hacer validaciones diferentes según el tipo de profesor. A estos subtipos se les agregó un atributo inferido, que se obtiene directamente de la transacción Profesor, y así se puede mostrar, por ejemplo, el nombre o email del docente sin guardar esos datos de nuevo.



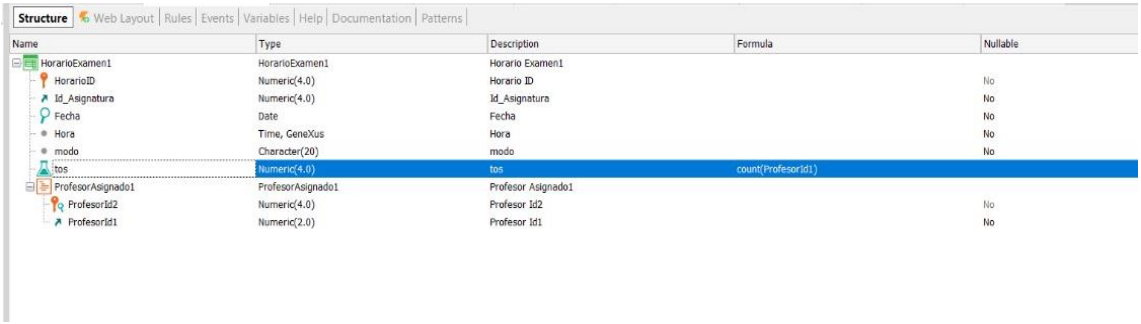
Transacciones con Segundo Nivel

Para representar la programación de exámenes dentro del sistema, se ha diseñado una transacción multinivel que permite reflejar una relación ordenada entre atributos.

En el primer nivel se almacenan los datos principales de la transacción, como su clave primaria (HorarioID), la fecha y la hora en la que se realizará el examen, incluyendo una clave foránea (Id_Asignatura) que nos permitirá obtener mejor información acerca del horario específico.

En el segundo nivel (nivel detalle), se incorporan los profesores que participarán, asignados mediante su clave primaria, lo que permitirá que el sistema no muestre otro nombre sin duplicar.

También se ha definido una fórmula TotalProfesores utilizando el siguiente cálculo con la función count(IdProfesor1), que permite contar los profesores que han sido asignados. Este atributo con fórmula se puede usar para una regla de validación, la cual exige que se asignen dos profesores, implementado en el apartado Rules.

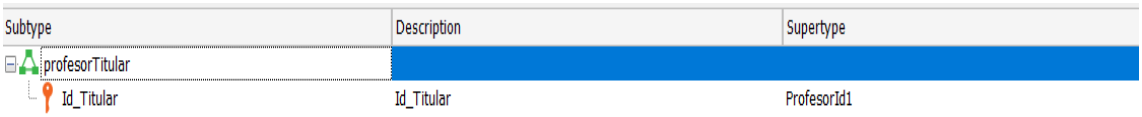


Justificación de Subtipos

Subtipo Profesor Titular

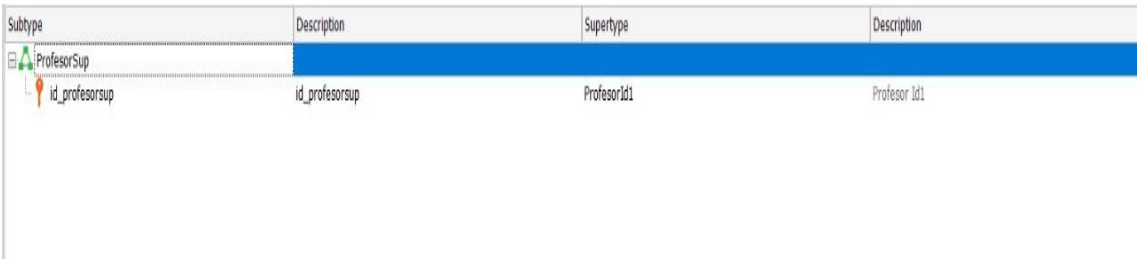
El profesor de subtipo fue creado para identificar a el profesor que tiene el rol principal y al docente que se encarga del rol secundario. Tomando como base ProfesorId1 y desde esa clave primaria generar el subtipo sin repetir ninguna estructura.

Esto nos sirve para la validación y que la lógica no sea igual que al de suplente y también se utilizo para mostrar los atributos de la transacción profesor sin necesidad de volverlos a guardar. Manteniendo una relación mas clara y entendible

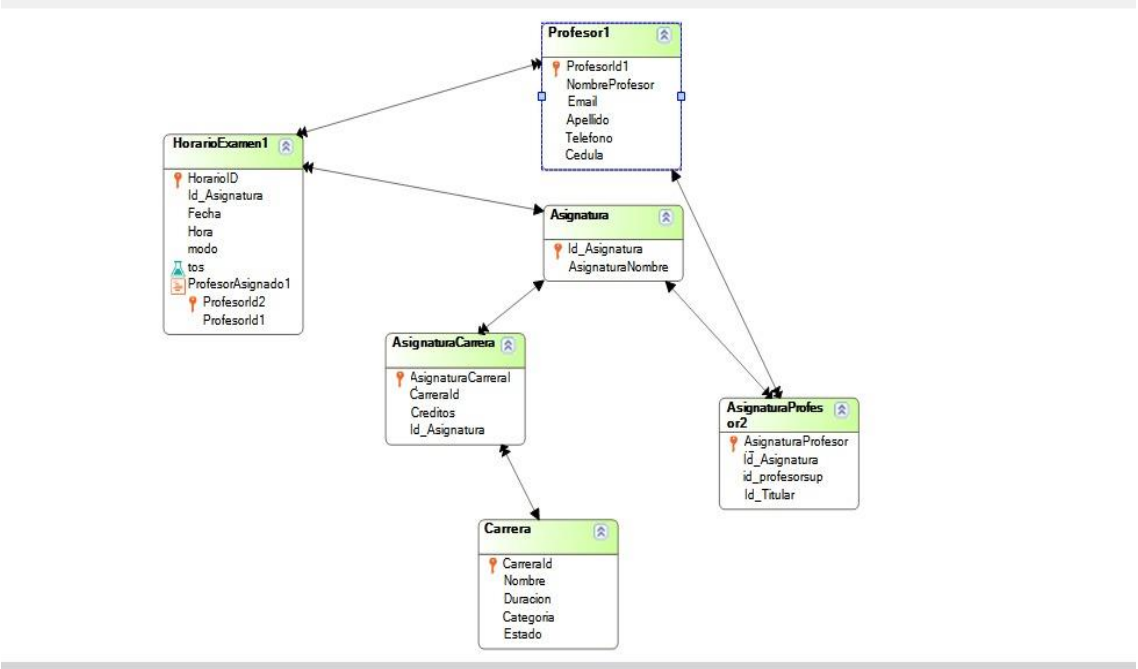


Subtipo ProfesorSuplente

El subtipo suplente también se creo con el fin de conocer quien ocupa el rol secundario, que de igual forma fue definido como subtipo de ProfesorId1 representando el rol alternativo en caso de la ausencia del titular, permitiendo modelar y validar como si fuera un atributo distinto de igual forma el titular pueden ser manipulados de forma independiente en reglas validaciones, aunque vengan de una misma clave primaria



Relaciones Generadas



La transacción AsignaturaProfesor2 controla las funciones docentes por materia, separando titular de suplente. Impide que varios profesores tengan el mismo cargo en una asignatura, creando una relación 1:N desde profesor hacia AsignaturaProfesor2. Un docente puede trabajar en múltiples materias con roles diferentes. Los subtipos usan el Profesor1para manejar distintos cargos sin duplicar estructuras.

HorarioExamen1 crea una relación N:M entre exámenes y profesores supervisores. Un examen puede tener varios supervisores y un profesor puede supervisar varios exámenes. GeneXus maneja esto automáticamente desde el detalle de la transacción, sin requerir tablas intermedias manuales.

El modelo organiza las relaciones con claves foráneas y preserva la integridad referencial. Carrera, Asignatura *y* Profesor son entidades independientes, mientras que AsignaturaCarrera, AsignaturaProfesor y el detalle de HorarioExamen son entidades

dependientes que necesitan otras entidades para existir, representando correctamente las conexiones del sistema.

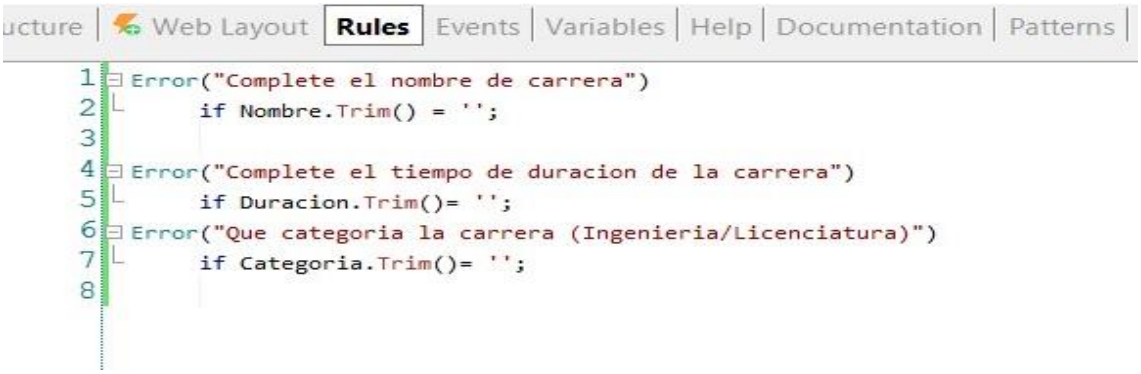
Validación de Campos Obligatorios y Advertencias

Para garantizar la integridad, la coherencia y la integridad ingresadas en cada transacción del sistema, se introdujeron las reglas de validación, realizadas automáticamente cuando se aprueban o almacenan. Estas validaciones, permiten que el no complemento evite los errores de registro de datos convencionales y mejore la lógica comercial definida para cada dispositivo. Las reglas de validación de Genexus son expresiones o condiciones configuradas en transacciones para verificar que los datos cumplan con ciertas reglas que implementa este software antes del almacenamiento en la base de datos. Esto evita incompleto, incorrecto o contrario al registro de los límites del modelo.

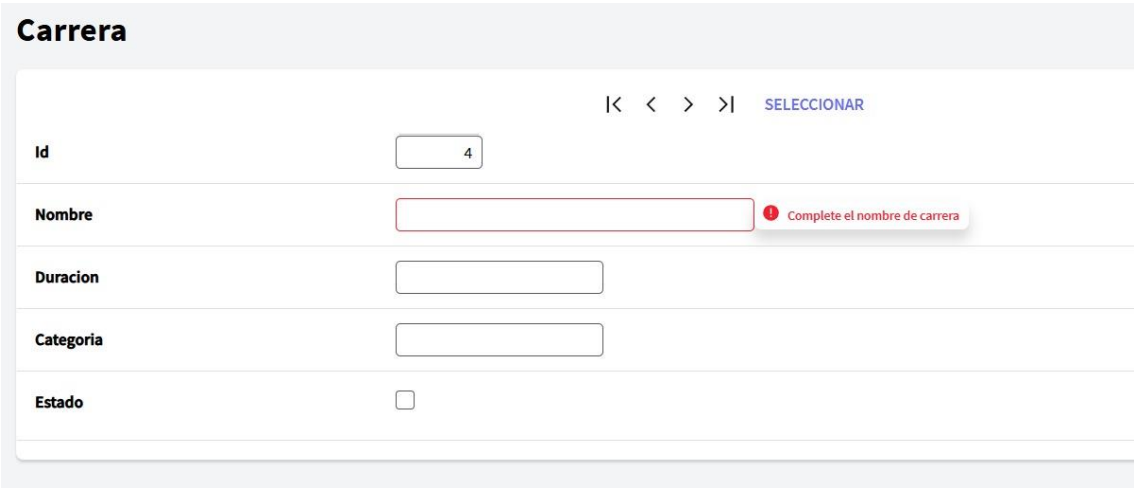
Se utilizaron las siguientes herramientas de validación para este sistema académico que nos permite implementarlo:

Transacción Carrera

Regla aplicada



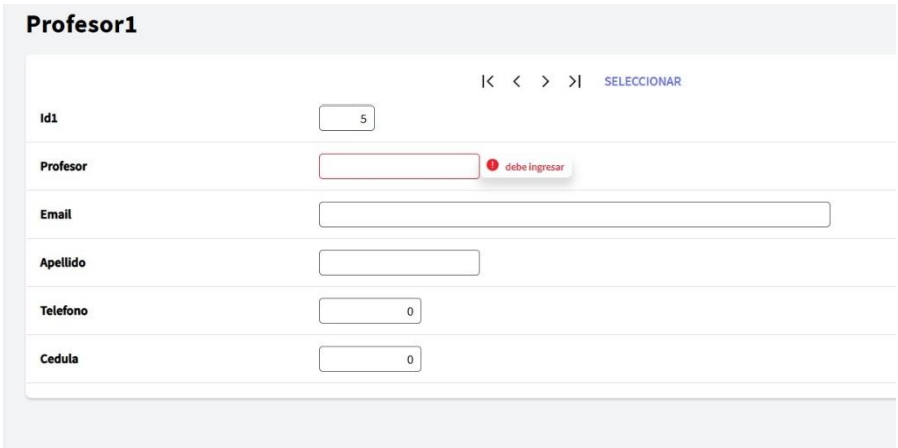
En esta regla se realiza la validación correspondiente para que no se deje ningún casillero vacío, utilizando la función Trim() en cada campo para asegurar que no se ingresen espacios en blanco.



Transacción Profesor

Reglas aplicadas

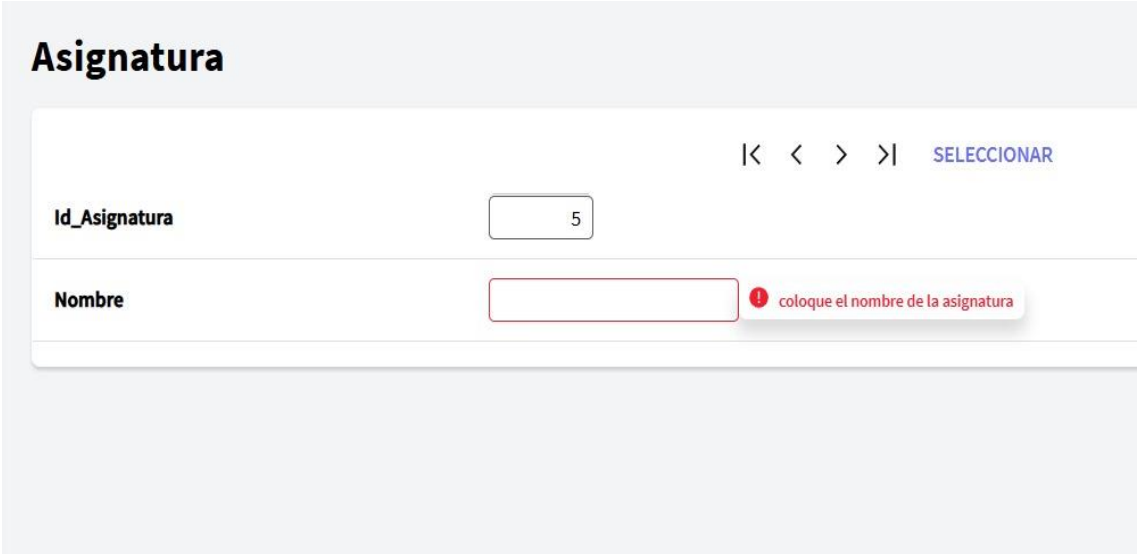
Con la función Trim() se valida los atributos de tipo carácter que no estén vacíos, y con la función IsEmpty() se verifica si los casilleros de los atributos tipo numérico no se encuentren vacíos, en esta validación se encuentra la validación que en caso de mandar un teléfono o cedula menor a 10 los datos no se guardaran en la base de datos



Transacción Asignatura

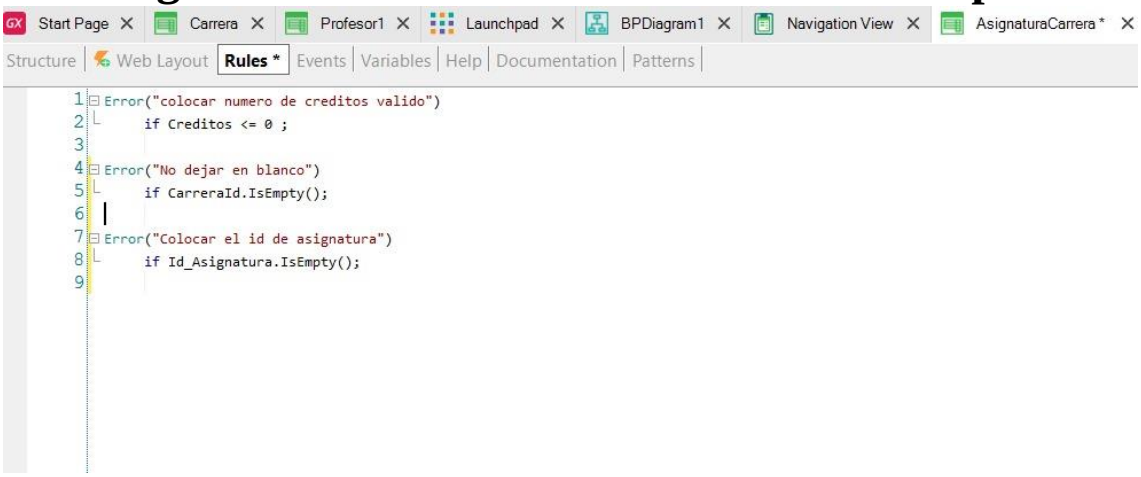
Regla aplicada

Al igual que en el anterior en esta regla también realiza el mismo proceso de no guardar los datos si un casillero esta vacío ni permitir espacios en blanco.

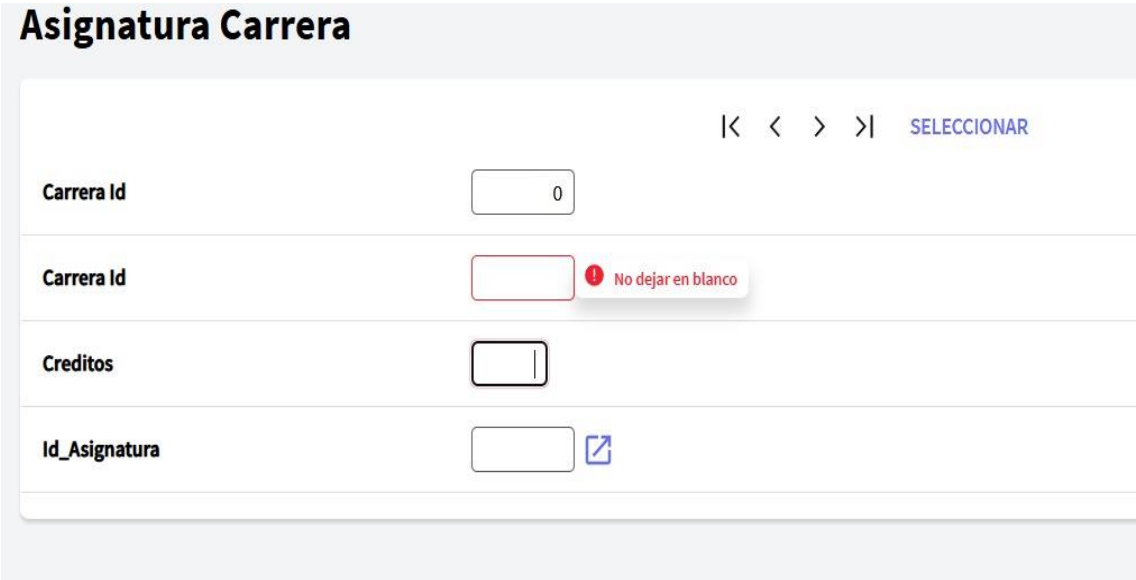


Transacción AsignaturaCarrera

Reglas aplicadas

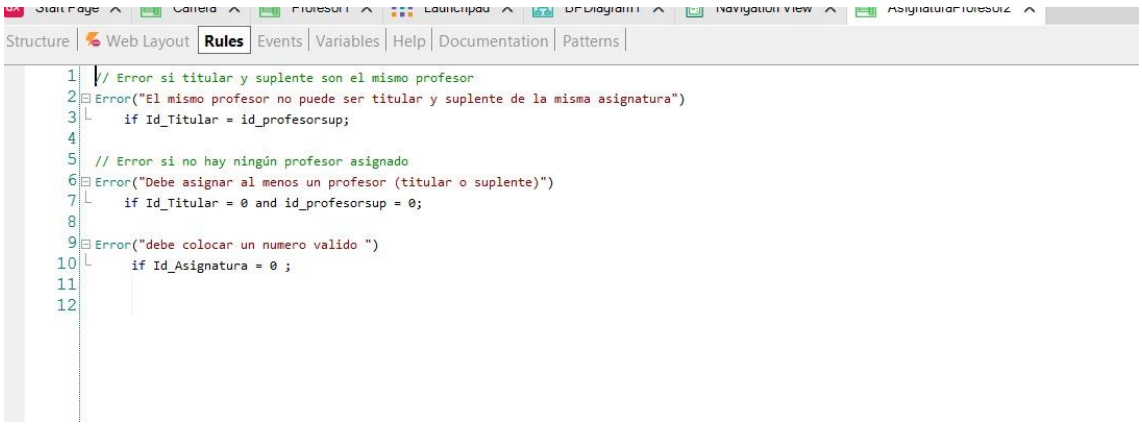


En esta regla se aplico la función IsEmpty() que nos permite validar números para que el casillero en donde el atributo sea numerico no se encuentre vacio en caso eso pues nos e guardara los datos.



Transacción AsignaturaProfesor

Reglas aplicadas

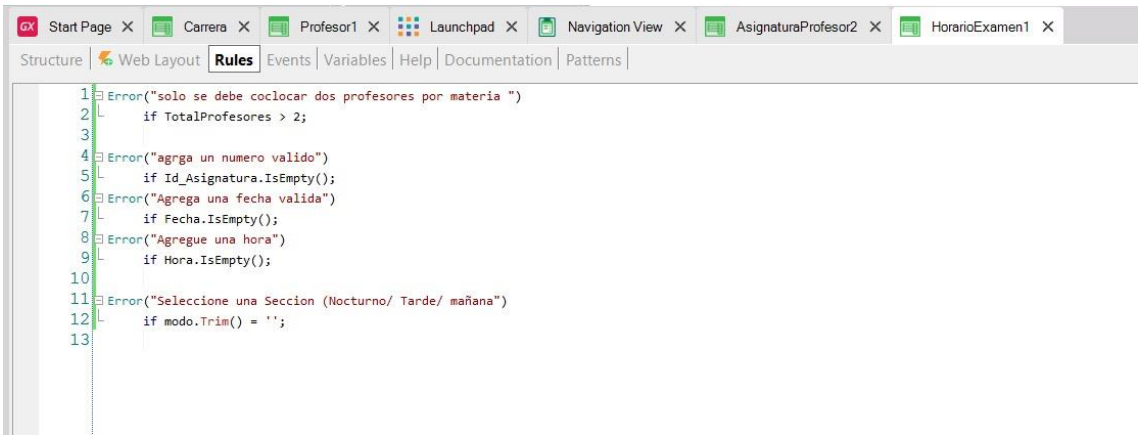


En esta validación se verifica que el profesor suplente no se encuentre también como profesor titular para evitar confuciones cuando se ingrese ala base de datos.

Transacción Horario Examen

Reglas aplicadas

Con la función IsEmpty() se verifica si los casilleros de los atributos tipo numérico no se encuentren vacíos. También incluye una validación que es basado en la formula que ya previamente se creo validando si mas 2 profesor están en una asignatura el programa no guardara los datos.

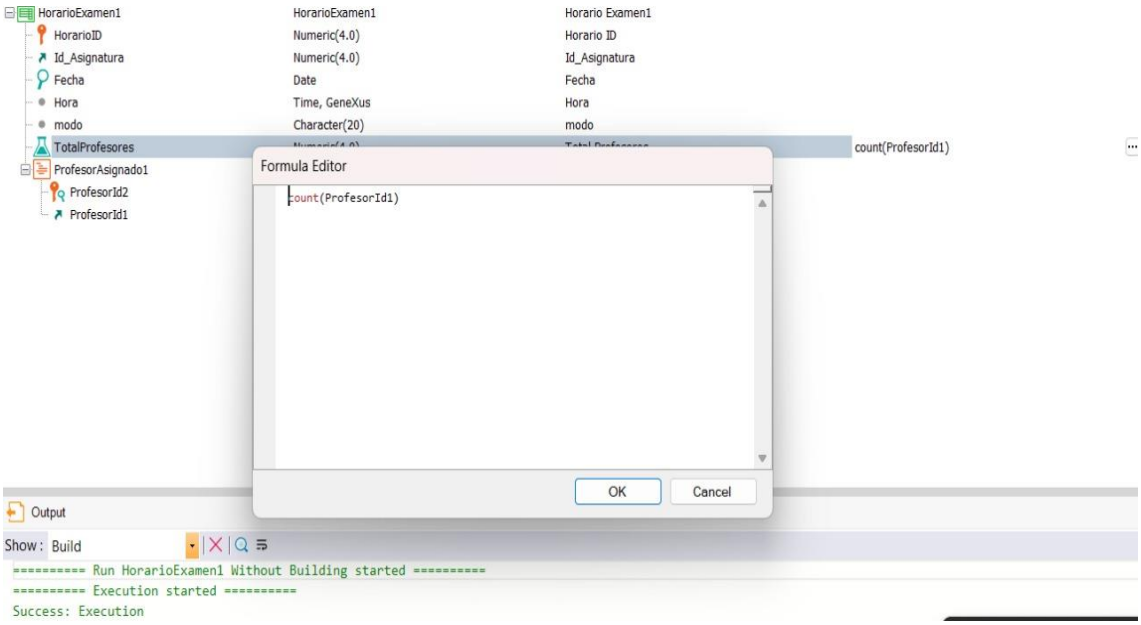


Justificación de Fórmulas

La fórmula TotalProfesores se usa para contar automáticamente cuántos profesores están asignados a un examen. Esta fórmula es clave en el sistema porque permite saber en el momento cuántos docentes están como supervisores, sin tener que contar manualmente y así se evitan errores.

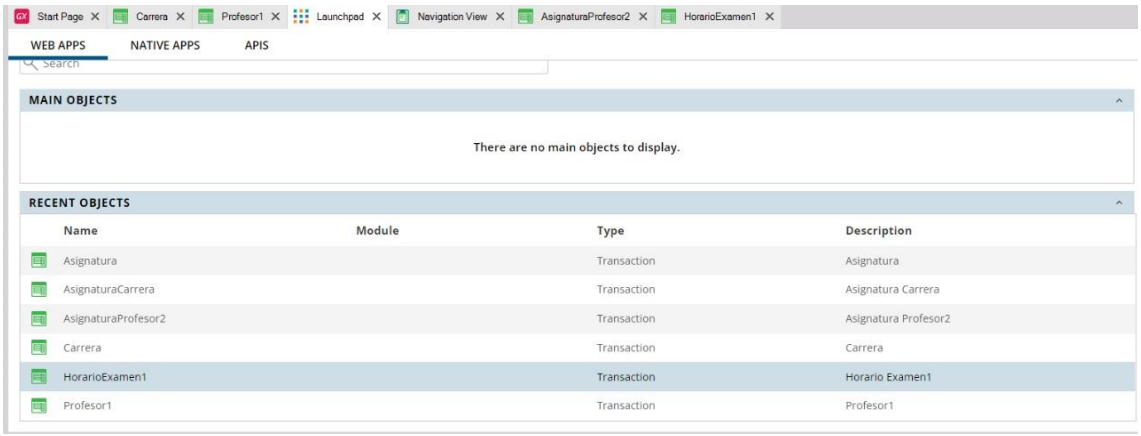
Se hizo como un atributo numérico calculado, activando la propiedad **Fórmula** y usando Count(ProfesorId) como expresión. Se le puso dos enteros porque no es común que haya más de 99 profesores en un solo examen.

Esta fórmula se encuentra en el primer nivel de la transacción HorarioExamen, y se actualiza sola cada vez que se agrega, se edita o se borra un profesor del segundo nivel. Con esto se asegura que el número siempre esté correcto según los cambios que se hagan.



Interfaz de usuario

Esta sección incluirá pantallas que ilustran tanto el proceso de prueba como la versión final del sistema utilizando una interfaz generada cuando se trabaja con un modelo web. Estas imágenes permiten la visualización del desarrollo y cómo las funciones de la interfaz web se estructuraron en diferentes etapas del proyecto.



Simulación de Datos en la Aplicación usando un prototipo

Transacción Asignatura (4 asignaturas)

Id_Asignatura	Nombre
0	Programacion
1	sistemas
2	Historia
3	Matematicas

CANCELAR

Transacción Carrera (3 carreras)

Id	Nombre	Duracion	Categoria	Estado
0				☐
1	software	8 semestre	ingenieria	☑
2	Educacion Fisica	8 semestres	Licenciatura	☑
3	Economia	9 semestres	Licenciatura	☑

Id1	Profesor	Apellido	Telefono	Cedula
0	juan	Matute	981874087	350319000
1	Justin	Hernandez	9818740871	1350319000
2	Emiro	Gomez	1245678912	1123456789
3	Adrian	Avila	981874087	1444444444
4	Juan	Taday	1234445555	5678904532

Transacción Profesor (5 profesores)

Transacción AsignaturaCarrera

Carrera Id	Creditos	Carrera Id	Id_Asignatura
1	4	0	0
2	6	2	1
3	5	3	2

Transacción AsignaturaProfesor

Profesor ID	Id_Asignatura	id_profesorsup	Id_Titular
0	1	4	1
1	1	4	1

Transacción Horario Examen

ID	Fecha	Hora	Id_Asignatura	modo
1	02/07/25	12:30	1	Nocturno
2	02/07/25	10:00	2	Nocturno

Integridad de datos

La integridad de la información en el sistema, está garantizada utilizando varios mecanismos clave:

- Una definición clara de la relación entre las transacciones utilizando claves primarias y extranjeras para conectar adecuadamente los dispositivos.
- Uso de reglas de validación que evitan que los datos incorrectos o incompletos obtengan la calidad del registro.
- La introducción del nivel secundario en las transacciones para modelar relaciones internas de una a muchos (1: n), al igual que los maestros asignados a cada examen.
- Control automático de la clave única de Genexus, que evita la creación de elementos duplicados o inconsistentes en la base de datos.

- Además, se integran los procedimientos (procedimientos), lo que le permite aprobar condiciones específicas antes de colocar o actualizar operaciones que fortalezcan la lógica de la empresa y garanticen el contexto de la información almacenada.

Conclusiones

En conclusión, con GeneXus pudimos hacer un sistema académico sólido usando la programación declarativa y bases de datos relacionales. La generación automática de código nos ayudó a crear la interfaz y la base de datos sin tener que programar todo a mano.

Las relaciones muchos a muchos y las fórmulas automáticas dieron estabilidad y un buen control de la información. Las validaciones, índices y el control de errores fueron importantes para asegurar que los datos estén bien y sin errores, algo fundamental en un sistema académico.

Por último, vimos que es necesario pensar en diferentes casos para hacer soluciones que sirvan de verdad y que la integridad de los datos se mantenga con claves foráneas y compuestas, evitando errores y asegurando que los datos relacionados estén bien conectados.

Anexos

Proyecto Generado:

https://drive.google.com/drive/folders/1_A5kYIEkf5ZUShvkhR2Gphdl2l2iGc7X?usp=sharing