UNIVERSIDAD ARGENTINA DE LA EMPRESA
FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS
CARRERA: ING. EN INFORMÁTICA | TEC. UNIVERSITARIA EN DESARROLLO DE SOFTWARE
ASIGNATURA: **Ingeniería de datos I**

Trabajo Práctico Integrador Obligatorio Final

Sistema de gestión de becados – Fundación BisBlick

Equipo de trabajo:
Grupo 7
- Jerez, Juan – 1152497
- Vilte, Vanina Solange – 1151215

Docente:
Franco Emanuel Salazar

FECHA DE PRESENTACIÓN: 26/06/2025

B. Resumen ejecutivo

Este informe documenta el desarrollo de un sistema de gestión de datos diseñado para optimizar el seguimiento académico, laboral y personal de los becados, la Fundación BisBlick necesitaba una solución estructurada para organizar y consultar información proveniente de diversos documentos y planillas.

El proyecto consistió en el diseño e implementación de una base de datos relacional sobre Microsoft SQL Server. Se partió de una hoja de cálculo real proporcionada por la fundación, a partir de la cual se construyó un modelo entidad-relación (DER) detallado, se aplicaron procesos de normalización y se creó un sistema que permite registrar, consultar y analizar los datos de los beneficiarios. El sistema contempla múltiples aspectos: situación académica, nivel de inglés, datos personales, intereses laborales, localización geográfica, entre otros.

Además del diseño e implementación de las tablas, se desarrollaron consultas funcionales, procedimientos almacenados (CRUD), triggers, vistas, funciones y una transacción controlada. También se incorporaron medidas de integridad, pruebas de validación, optimización de rendimiento mediante índices y estimaciones presupuestarias simuladas.

El proyecto se realizó en equipo bajo una metodología ágil (SCRUM) con roles y tareas definidos. Este trabajo no solo resuelve una necesidad concreta de la fundación, sino que también demuestra la capacidad del equipo para aplicar conocimientos técnicos a una situación real, con impacto social.

C. Introducción

El presente trabajo práctico final tiene como objetivo aplicar los conceptos fundamentales de bases de datos relacionales. Bisblick es una organización sin fines de lucro, que promueve la igualdad de oportunidades educativas a través de programas de acompañamiento académico, económico y humano para jóvenes de alto potencial.

El propósito de este proyecto es crear una solución tecnológica sólida y escalable que ayude a la organización a optimizar el seguimiento de sus beneficiarios y a mejorar la gestión de sus programas.

El sistema abarca la administración de datos de jóvenes, mentores, programas y beneficios, lo que permitirá un acceso eficiente a la información y facilitará la toma de decisiones estratégicas por parte del equipo de la ONG.

D. Objetivos del sistema

Objetivo general

Diseñar e implementar un sistema de base de datos relacional para la Fundación BisBlick que permita registrar, consultar, actualizar y analizar la información integral de los becados, facilitando el seguimiento académico, laboral y social de cada uno de ellos.

Objetivos específicos

Desarrollar una estructura de datos robusta que contemple las relaciones entre jóvenes, mentores, capacitaciones, becas y programas laborales.
Implementar mecanismos de seguridad y control de acceso que aseguren la confidencialidad de los datos.
Facilitar la generación de reportes e informes que permitan evaluar el impacto de las acciones de la ONG.
Permitir que el sistema sea escalable para acompañar el crecimiento de la organización.

Requisitos funcionales

- Registro, edición y eliminación de datos de jóvenes, mentores y programas.
- Asociación de beneficiarios con becas, capacitaciones y mentorías.
- Generación de informes de seguimiento e impacto.

Requisitos no funcionales

- Seguridad en el manejo de datos sensibles.
- Escalabilidad para acompañar el crecimiento de la base de datos.
- Compatibilidad con tecnologías ampliamente utilizadas en entornos corporativos (como SQL Server).

E. Diseño de base de datos

Diseño Conceptual - Diagrama entidad-relación

El diagrama entidad-relación (DER) del sistema de gestión para Fundación BisBlick ofrece una representación completa y bien estructurada de la información sobre los estudiantes becados, abarcando su situación académica y laboral, la conexión con sus mentores, y los informes de seguimiento que se generan a lo largo del tiempo.
Este modelo facilita la gestión integral del ciclo de vida del becado dentro del programa, asegurando coherencia, trazabilidad y apoyo en la toma de

decisiones. Las entidades y relaciones están organizadas en ocho áreas funcionales, que operan de manera interconectada y cumplen con las reglas de integridad referencial, garantizando así la consistencia de los datos.

Región de becados

Representa a las personas que forman parte del programa de becas.

Entidad: Becado

Identificada por ID_Becado (PK), contiene los siguientes atributos personales y de contexto:

- **Nombre, Apellido**: datos personales.
- **DNI**: identificador oficial.
- **Fecha_Nacimiento, Género**.
- **Dirección, Localidad_Actual, Provincia_Actual**: Información de la dirección del becado.
- **ONG**: organización de referencia.
- **Tiene_Cobertura_Medica**: puede ser 'Sí' o 'No'.
- **Nombre_Cobertura_Medica**: nombre de la obra social o prepaga, si aplica.

Relaciones importantes:

- “Tiene” con DatosLaborales (1:1): Cada becado tiene datos laborales cargados.
- “Cuenta con” con Contacto (1:1): Relación exclusiva con sus medios de contacto.
- “Posee” con DatosAcademicos (1:1): Cada becado tiene sus datos académicos cargados..
- “Tiene” con Beca (0:N): El becado puede o no tener una beca.
- “Recibe” con Informe-Becado (1:N): Relación que permite registrar informes institucionales por parte del equipo de seguimiento.
- “Guiado por” con Mentor (0:1): un becado puede o no tener un mentor.

Región de becas

Representa los programas de becas que se otorgan.

Entidad: Beca

Identificada por ID_Beca (PK), incluye:

- **Año_Inicio, Mes_Inicio**: fecha de inicio.
- **Tipo_Beca**: puede ser 'Estudio' o 'Trabajo'.
- **Estado**: puede tomar los valores 'Activa', 'Finalizada', 'Egresado' o 'Abandono'.
- **Modalidad**: modalidad de participación en la ONG.
- **Año_Cambio**: año en que pasó a otra modalidad.

- **Año_Abandono**: si aplica.
- **Año_Egreso**: año de finalización regular(si aplica).
- **Licencia_ingles**: 'Sí' o 'No' según acceso a la plataforma de idiomas.
- **Pedido_PC**: 'Sí' o 'No', si pidió equipo tecnológico.
- **Empresa**: referencia a la empresa que otorga la beca.

Relación:

- **“Tiene”** con Becado (0:N):
  Un becado puede tener ninguna, una o varias becas asignadas a lo largo del tiempo. Esta relación permite registrar la evolución de cada persona dentro del programa.

Región académica

Esta región representa la trayectoria educativa del becado.

Entidad: DatosAcademicos
Identificada por ID_Academico (PK), contiene:

- **Carrera**: nombre de la carrera cursada.
- **Universidad**: institución en la que cursa.
- **Año_Ingreso**: año en que ingresó a la carrera.
- **Tipo**: indica si la institución es 'Privada' o 'Pública'.
- **Beca_Universidad**: Si es privada, si recibe algun tipo de beca.
- **Area**: puede incluir valores como 'Grado', 'Pregrado', 'Corta Duración', entre otros.
- **Nivel_Ingles**: nivel determinado o evaluado, con posibles valores como 'Básico', 'Intermedio', 'Avanzado'.
  Examenes_Ingles: tipo de certificación (por ejemplo, 'TOEFL', 'IELTS', 'First Certificate' o 'Ninguno').

Relación:

- “Posee” con becado (1:1) Cada Becado posee un único conjunto de datos académicos.

Región de contacto

Contiene los medios de comunicación del becado.

Entidad: Contacto
Identificada por ID_Contacto (PK):

- **Telefono**: número principal.
- **Telefono_Alternativo**: segundo número opcional.
  Mail: correo electrónico personal.

Relación:

- **“Contactado mediante”** con Becado (1:1):
  Cada becado tiene un único registro de contacto que incluye teléfono, mail y número alternativo. La relación es uno a uno y exclusiva.

Región laboral

Modela la situación laboral del becado.

Entidad: DatosLaborales
Identificada por ID_Laboral (PK), incluye:

- **Ocupación**: Si trabaja o no. puede responderse “Si” o “No”.
- **Ubicacion, Puesto** : Información de su trabajo.
- **CargaHoraria**: Puede completarse con “FullTime”,”PartTime” o ”Rotativo”.
- **Busqueda-Laboral**: 'Sí' o 'No'.
- **LinkedIn**: enlace al perfil, si tiene.

Relación:
“Tiene” con Becado (1:1):
Cada becado tiene un único conjunto de datos laborales. La relación es directa y de tipo uno a uno. Puede no estar completa en algunos casos, pero se reserva el espacio para su eventual carga.

Región de mentores

Representa a los profesionales encargados de acompañar y guiar a los becados a lo largo de su desarrollo académico y personal.

Entidad: Mentor
Identificada por ID_Mentor (PK), incluye:

- **Nombre, DNI**: datos identificatorios.
- **Carrera**: formación profesional del mentor.

- **Puesto**: cargo actual en su lugar de trabajo.
- **Ubicación**: lugar de trabajo o residencia profesional.
- **Mail**: dirección de correo electrónico.
- **Teléfono**: número de contacto.
- **Estado**: puede ser ‘Activo’ o ‘Inactivo’, según su participación actual.

Relación:

- Guiado por (0:1): cada becado puede estar guiado por un mentor, aunque no es obligatorio.
- Genera (1:N): relación con **Informe-Mentor**, que permite registrar múltiples informes de seguimiento sobre su actividad con los becados.

Región de Informes

Permite registrar el seguimiento sistemático de los becados, tanto desde la perspectiva institucional como desde la mentoría.

Entidad: Informe-Becado
Identificada por ID_Informe (PK), incluye:

- **Informe_Marzo_Abril**
- **Informe_Julio_Agosto**
- **Informe_Noviembre_Diciembre**
- **Comentario**: observaciones generales del seguimiento.

Relación:

- Realiza (1:N): un becado puede tener múltiples informes institucionales a lo largo del año.

Entidad: Informe-Mentor
Identificada también por ID_Informe (PK), incluye:

- **Informe_Marzo_Abril**
- **Informe_Julio_Agosto**
- **Informe_Noviembre_Diciembre**
- **Comentario**: apreciaciones y evaluación del mentor sobre el progreso del becado.

Relación:

- Genera (1:N): cada mentor puede generar múltiples informes de seguimiento sobre los becados que tiene a cargo.

Diseño lógico - Modelo relacional

El diseño lógico del sistema es la conversión directa del modelo conceptual (DER) a un modelo relacional que se puede implementar en SQL Server. En esta fase, las entidades se convierten en tablas relacionales, sus atributos se transforman en columnas con los tipos de datos adecuados, y las relaciones entre ellas se establecen mediante claves primarias y foráneas, siguiendo las reglas de cardinalidad y participación del modelo conceptual.

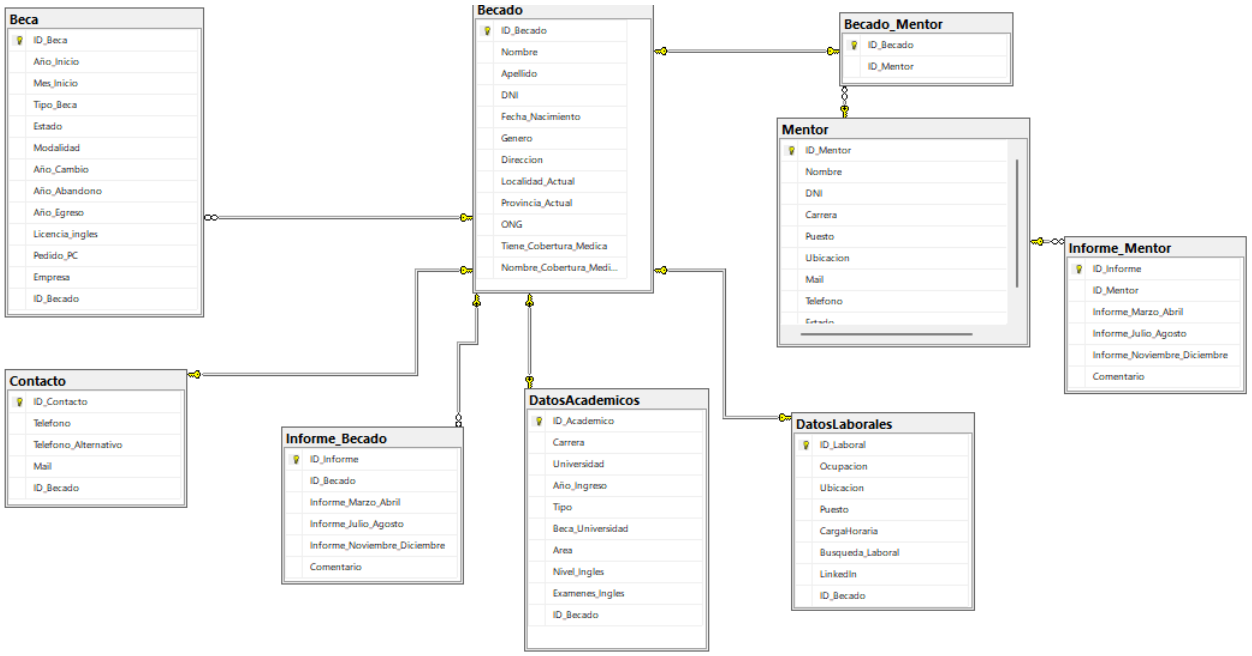
El modelo relacional resultante organiza de manera estructurada la información clave del sistema, distribuida en ocho áreas funcionales, garantizando la integridad referencial, la consistencia en el almacenamiento y una base sólida para futuras consultas y mantenimientos.

Este modelo presenta las siguientes características:

- **Claves primarias** que aseguran la unicidad de cada registro.
- **Claves foráneas** que mantienen la conexión lógica entre entidades relacionadas.
- Restricciones **CHECK** para validar los valores permitidos (como 'Sí' / 'No', o categorías específicas).
- Campos únicos (UNIQUE) en relaciones uno a uno.

Normalización hasta al menos la tercera forma normal (**3FN**), eliminando redundancias y asegurando que cada atributo dependa únicamente de la clave primaria.

El modelo relacional, diseñado en SQL Server, se ilustra gráficamente en el siguiente diagrama lógico:



Diseño físico - Implementación en SQL Server

El diseño físico representa la fase final del proceso de modelado de la base de datos, donde el modelo lógico es traducido directamente al sistema gestor de bases de datos elegido, en este caso SQL Server. En esta etapa, se crean las tablas, se asignan los tipos de datos más apropiados, se definen claves primarias y foráneas, restricciones de integridad y configuraciones necesarias para garantizar el correcto almacenamiento y funcionamiento del sistema.

El modelo relacional fue implementado mediante sentencias **CREATE TABLE** en SQL Server, respetando los siguientes criterios:

- Claves primarias (**PRIMARY KEY**) para identificar de forma única cada registro.
- Claves foráneas (**FOREIGN KEY**) para establecer las relaciones entre las entidades y asegurar la integridad referencial.
- Restricciones **CHECK** para validar valores permitidos en campos específicos, como los que aceptan solo 'Si' o 'No', o listas cerradas como 'Activo' / 'Inactivo'.
- Campos **UNIQUE** en relaciones uno a uno, como en Contacto, DatosAcademicos o DatosLaborales, que dependen exclusivamente de un becado.

Tipos de datos adecuados a la naturaleza del campo, por ejemplo:

- **VARCHAR**(n) para textos y cadenas alfanuméricas.
- **DATE** para fechas de nacimiento.
- INT con **IDENTITY** para generar identificadores automáticos en Becado.

Además, se crearon tablas intermedias, como **Becado_Mentor**, para modelar relaciones como "un becado puede o no tener un mentor" sin perder la flexibilidad del modelo ni comprometer la integridad de los datos.

Este diseño físico garantiza que la base de datos pueda ser implementada de forma directa y funcional, respetando las reglas definidas en el modelo conceptual y lógico, y facilitando tanto la carga de datos como las futuras consultas, informes y mantenimientos del sistema.

F. Dependencia funcional

En bases de datos relacionales, una dependencia funcional describe la relación entre dos conjuntos de atributos de una tabla. Un atributo **A** determina funcionalmente a otro atributo **B** si, para cada valor de **A**, existe un único valor de **B**. Se denota como **A** → **B**. Es decir, "**B** depende funcionalmente de **A**".
Este concepto es clave para identificar redundancias, detectar posibles errores lógicos y llevar una base de datos a su forma normal correspondiente.

Aplicación en el modelo BisBlick
Durante el análisis del modelo de la Fundación BisBlick, se identificaron múltiples dependencias funcionales relevantes que permitieron normalizar correctamente la base de datos hasta la tercera forma normal (3FN).

Tabla Becado

- ID_Becado → Nombre, Apellido, DNI, Fecha_Nacimiento, Genero, Direccion, Localidad_Actual, Provincia_Actual, ONG, Tiene_Cobertura_Medica, Nombre_Cobertura_Medica
- Esto significa que el identificador del becado determina de manera única toda su información personal y contextual.

Tabla DatosAcademicos

- ID_Becado → Carrera, Universidad, Año_Ingreso, Tipo, Beca_Universidad, Area, Nivel_Ingles, Examenes_Ingles
- Cada becado tiene una única trayectoria académica registrada, por lo que basta con su ID para recuperar todos los datos educativos.

Tabla Mentor

- ID_Mentor → Nombre, DNI, Carrera, Puesto, Ubicacion, Mail, Telefono, Estado
- Cada mentor está completamente definido por su ID, sin necesidad de duplicar sus datos por cada informe o becado asociado.

Tabla Informe_Becado

- ID_Informe → ID_Becado, Informe_Marzo_Abril, Informe_Julio_Agosto, Informe_Noviembre_Diciembre, Comentario
- Cada informe tiene una relación directa y única con el becado y los datos de seguimiento correspondientes.

Tabla Contacto

- ID_Contacto → Telefono, Telefono_Alternativo, Mail, ID_Becado
- Cada contacto está vinculado a un solo becado. Gracias a la restricción UNIQUE sobre ID_Becado, se mantiene la relación 1:1 y se evita duplicación de medios de contacto.

Tabla DatosLaborales

- ID_Laboral → Ocupacion, Ubicacion, Puesto, CargaHoraria, Busqueda_Laboral, LinkedIn, ID_Becado
- Permite registrar en un solo lugar la situación laboral de un becado, respetando la unicidad y evitando inconsistencias entre campos laborales.

Tabla Beca

- ID_Beca → Año_Inicio, Mes_Inicio, Tipo_Beca, Estado, Modalidad, Año_Cambio, Año_Abandono, Año_Egreso, Licencia_ingles, Pedido_PC, Empresa, ID_Becado
- Cada beca tiene un conjunto único de atributos relacionados. Esto permite que un becado tenga varias becas a lo largo del tiempo, sin mezclar o sobrescribir datos previos.

Tabla Becado_Mentor

- ID_Becado → ID_Mentor
- Cada becado puede tener a lo sumo un mentor, lo cual se representa mediante esta tabla intermedia. La clave primaria ID_Becado evita asignar más de un mentor simultáneamente.

Tabla Informe_Mentor

- ID_Informe → ID_Mentor, Informe_Marzo_Abril, Informe_Julio_Agosto, Informe_Noviembre_Diciembre, Comentario
- Registra múltiples informes de un mismo mentor sin duplicar información innecesaria, manteniendo la trazabilidad de su participación en el programa.

Bondades de aplicar dependencias funcionales correctamente

Aplicar correctamente las dependencias funcionales permite:

- Evitar redundancia de datos (por ejemplo, no repetir carrera y universidad en varias tablas).
- Prevenir inconsistencias (ej: si cambia la universidad de un becado, no hay que modificarla en varios lugares).
- Hacer más eficientes las actualizaciones y consultas.
- Facilitar la integridad referencial y la trazabilidad de los registros.
- Preparar una base sólida y escalable para futuros módulos o sistemas.

H. Plan de desarrollo

Para organizar el desarrollo del sistema se utilizó la metodología ágil **SCRUM**. Esta metodología se caracteriza por dividir el trabajo en entregas parciales e iterativas llamadas sprints, en las cuales cada integrante del equipo asume un rol activo y responsabilidades concretas. El objetivo es fomentar la colaboración, la adaptación continua y la entrega progresiva de valor.

Cronograma de trabajo

Sprint	**Objetivo principal**	**Entregables / Actividades clave**
Sprint 1	Análisis y diseño del modelo	Análisis de las hojas de cálculo, diseño del DER y modelo lógico
Sprint 2	Implementación técnica de la base de datos	Scripts CREATE, carga de datos reales, diseño físico en SQL Server
Sprint 3	Desarrollo de funciones, consultas y documentación	Procedimientos, vistas, triggers, función, transacción y documento
Sprint 4	Validación, ajustes y presentación final	Revisión integral, presentación e informe final

Herramientas utilizadas

- Microsoft SQL Server Management Studio (SSMS): Para implementar la base de datos, ejecutar scripts y consultas.
- draw.io: Para diseñar el Diagrama Entidad-Relación (DER).
- Google Docs: Para redactar el documento final.
- Canva: Para preparar la presentación.
- Google Sheets: Para organizar los datos de las planillas que nos proporcionó Bisblick y estructurar registros de prueba.

J. Consultas SQL

Las consultas que realizamos fueron las siguientes

Consultas básicas

Estas consultas realizan operaciones simples de selección, filtrado, agrupación y ordenación de datos para obtener información básica del sistema.

- Conteo de becados por universidad:

```sql
SELECT Universidad, COUNT(*) AS Cantidad_Becados --cantidad de becados de bisblick por universidad
FROM DatosAcademicos
GROUP BY Universidad;
```

- Conteo por ocupación actual:

```sql
SELECT Ocupacion, COUNT(*) AS Total --si actualmente trabaja o no.
FROM DatosLaborales
GROUP BY Ocupacion;
```

- Conteo de becados por género y búsqueda laboral:

```sql
SELECT B.Genero, DL.Busqueda_Laboral, COUNT(*) AS Total --si esta en busqueda laboral, porcentaje de genero
FROM DatosLaborales DL
JOIN Becado B ON DL.ID_Becado = B.ID_Becado
GROUP BY B.Genero, DL.Busqueda_Laboral;
```

- Cantidad de becados por provincia:

```sql
SELECT Provincia_Actual, COUNT(*) AS Cantidad --cantidad de becados por provincia
FROM Becado
GROUP BY Provincia_Actual;
```

- Nivel de inglés por nombre del becado:

```
SELECT B.Nombre, B.Apellido, DA.Nivel_Ingles
FROM Becado B
JOIN DatosAcademicos DA ON B.ID_Becado = DA.ID_Becado;
```

- Becados por universidad:

```
SELECT DA.Universidad, COUNT(*) AS Cantidad_Becados
FROM DatosAcademicos DA
JOIN Becado B ON DA.ID_Becado = B.ID_Becado
GROUP BY DA.Universidad;
```

- Rango de edades:

```
SELECT --rango de edades
    CASE
        WHEN Edad BETWEEN 16 AND 18 THEN '16-18'
        WHEN Edad BETWEEN 19 AND 22 THEN '19-22'
        WHEN Edad BETWEEN 23 AND 26 THEN '23-26'
        WHEN Edad BETWEEN 27 AND 30 THEN '27-30'
        ELSE '31+'
    END AS Rango_Edad,
    COUNT(*) AS Cantidad
FROM (
    SELECT DATEDIFF(YEAR, Fecha_Nacimiento, GETDATE()) AS Edad
    FROM Becado
) AS SubEdad
GROUP BY
    CASE
        WHEN Edad BETWEEN 16 AND 18 THEN '16-18'
        WHEN Edad BETWEEN 19 AND 22 THEN '19-22'
        WHEN Edad BETWEEN 23 AND 26 THEN '23-26'
        WHEN Edad BETWEEN 27 AND 30 THEN '27-30'
        ELSE '31+'
    END;
```

Vistas, procedimientos y funciones:
Para facilitar el acceso, manipulación y validación de datos, el sistema incluye:

- **Vistas**: Consultas predefinidas que combinan y simplifican la información compleja para un uso más ágil, por ejemplo:
    - **Vista_InfoCompletaBecado**: vista con información combinada de becado, académica, laboral y beca.

```
CREATE VIEW Vista_InfoCompletaBecado AS
SELECT
    B.Nombre, B.Apellido, B.Genero, B.Provincia_Actual,
    DA.Universidad, DA.Carrera, DL.Ocupacion, DL.CargaHoraria,
    Beca.Estado, Beca.Tipo_Beca
FROM Becado B
LEFT JOIN DatosAcademicos DA ON B.ID_Becado = DA.ID_Becado
LEFT JOIN DatosLaborales DL ON B.ID_Becado = DL.ID_Becado
LEFT JOIN Beca ON B.ID_Becado = Beca.ID_Becado;
GO
```

    - **Vista_BecasPorEmpresa**: resumen de la cantidad de becas por empresa.

```
CREATE VIEW Vista_BecasPorEmpresa AS
SELECT Empresa, COUNT(*) AS Cantidad_Becas
FROM Beca
GROUP BY Empresa;
GO
```

- **Triggers**: Automatizan validaciones para asegurar la integridad de los datos, tales como:
    - **tr_ValidarEstadoBeca:** valida que el estado de la beca sea uno válido.

```
CREATE TRIGGER tr_ValidarEstadoBeca
ON Beca
AFTER INSERT
AS
BEGIN
    IF EXISTS (
        SELECT *
        FROM INSERTED
        WHERE Estado NOT IN ('Activa', 'Finalizada', 'Egresado', 'Abandono')
    )
    BEGIN
        RAISERROR('El estado ingresado para la beca no es válido.', 16, 1);
        ROLLBACK;
    END
END;
GO
```

- **tr_ValidarEdadBecado:** verifica que los becados tengan al menos 16 años.

```
CREATE TRIGGER tr_ValidarEdadBecado
ON Becado
AFTER INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM INSERTED
        WHERE DATEDIFF(YEAR, Fecha_Nacimiento, GETDATE()) < 16
    )
    BEGIN
        RAISERROR('El becado debe tener al menos 16 años.', 16, 1);
        ROLLBACK;
    END
END;
GO
```

- **Procedimientos almacenados:** bloques de código reutilizables para insertar datos con validaciones encapsuladas:

  - **sp_InsertarBecado:** procedimiento para insertar un nuevo becado con sus datos básicos.

```
CREATE PROCEDURE sp_InsertarBecado
    @Nombre VARCHAR(50),
    @Apellido VARCHAR(50),
    @DNI VARCHAR(20),
    @Fecha_Nacimiento DATE,
    @Genero VARCHAR(20),
    @Direccion VARCHAR(100),
    @Localidad_Actual VARCHAR(50),
    @Provincia_Actual VARCHAR(50),
    @ONG VARCHAR(100),
    @Tiene_Cobertura_Medica VARCHAR(2),
    @Nombre_Cobertura_Medica VARCHAR(100)
AS
BEGIN
    INSERT INTO Becado (
        Nombre, Apellido, DNI, Fecha_Nacimiento, Genero,
        Direccion, Localidad_Actual, Provincia_Actual,
        ONG, Tiene_Cobertura_Medica, Nombre_Cobertura_Medica
    )
    VALUES (
        @Nombre, @Apellido, @DNI, @Fecha_Nacimiento, @Genero,
        @Direccion, @Localidad_Actual, @Provincia_Actual,
        @ONG, @Tiene_Cobertura_Medica, @Nombre_Cobertura_Medica
    );
END;
GO
```

- Funciones definidas por el usuario: para cálculos o verificaciones específicas:
    - **fn_TieneBecaActiva:** función que retorna si un becado tiene alguna beca activa.

```
CREATE FUNCTION fn_TieneBecaActiva (
    @ID_Becado INT
)
RETURNS VARCHAR(2)
AS
BEGIN
    DECLARE @Resultado VARCHAR(2);

    IF EXISTS (
        SELECT 1
        FROM Beca
        WHERE ID_Becado = @ID_Becado AND Estado = 'Activa'
    )
        SET @Resultado = 'Si';
    ELSE
        SET @Resultado = 'No';

    RETURN @Resultado;
END;
GO
```

K. Transacciones

En el desarrollo del sistema de gestión para la Fundación BisBlick, se implementaron transacciones como un mecanismo fundamental para garantizar que las operaciones en la base de datos se realicen de manera segura, coherente y controlada. Las transacciones permiten agrupar múltiples instrucciones SQL en una única unidad lógica de trabajo, que se ejecuta en su totalidad o no se ejecuta en absoluto, respetando las propiedades ACID:

Atomicidad: asegura que todas las acciones dentro de una transacción se completen por completo o no se realicen en absoluto. Por ejemplo, al registrar un nuevo becado, si se ingresan sus datos personales, académicos y laborales, todas las inserciones deben llevarse a cabo correctamente. Si hay un error en alguna de ellas, todo el proceso se revierte, evitando registros parciales o inconsistentes.

Consistencia: garantiza que la base de datos pase de un estado válido a otro también válido, cumpliendo con las reglas de integridad definidas (como claves primarias, foráneas o restricciones CHECK). En nuestro sistema, esto previene que se asignen mentores que no existen o becas con valores inválidos.

Aislamiento: impide interferencias entre transacciones que se ejecutan al mismo tiempo. Si varios usuarios acceden o modifican los datos de un mismo becado, el aislamiento asegura que cada transacción trabaje con una vista coherente de los datos, evitando efectos colaterales hasta que se completen.

Durabilidad: una vez que se confirma una transacción con COMMIT, los cambios se guardan de forma permanente, incluso si hay fallos en el sistema. Esto es crucial para garantizar que operaciones como el registro de un informe o la actualización del estado de una beca no se pierdan.
Durante la implementación en SQL Server, se utilizaron instrucciones como BEGIN TRANSACTION, COMMIT y ROLLBACK para gestionar estas operaciones críticas, asegurando así un comportamiento confiable y predecible del sistema frente a errores o accesos simultáneos.

L. optimización y rendimiento

Una base de datos no solo debe ser funcional y correcta, sino también eficiente. La optimización busca mejorar los tiempos de respuesta de las consultas y reducir la carga sobre el sistema, especialmente a medida que crece el volumen de datos.
En el proyecto para la Fundación BisBlick se aplicaron estrategias básicas de optimización que permiten escalar el sistema en el tiempo y mantener un buen rendimiento de lectura y escritura.

Índices implementados

Se crearon índices sobre columnas clave que son comúnmente utilizadas en cláusulas WHERE, JOIN o GROUP BY, con el fin de acelerar búsquedas y ordenamientos.
Si quisiéramos aplicar un índice en BisBlick podríamos usarlo, por ejemplo, en:

- Fecha_Nacimiento (para consultas de edad)
- Carrera (en DatosAcademicos, para búsquedas por área)
- Provincia (en Becado, para agrupamiento geográfico)

Supongamos que hacemos muchas consultas por provincia:

```
CREATE INDEX idx_becado_provincia ON Becado(Provincia);
```

Esto optimiza esta consulta:

```
SELECT Provincia, COUNT(*)
FROM Becado
GROUP BY Provincia;
```

Consultas optimizadas

Durante el desarrollo se aplicaron buenas prácticas para optimizar las consultas:

- Se usaron SELECT específicos (con columnas) en lugar de SELECT * para evitar traer datos innecesarios.

- Se priorizó el uso de JOIN explícito en lugar de subconsultas anidadas donde era más eficiente.
- Se agregaron filtros WHERE para limitar resultados en lugar de procesar toda la tabla.

Gestión del rendimiento en SQL Server

SQL Server optimiza internamente muchas consultas usando su motor de ejecución, planificando automáticamente el orden de ejecución, uso de índices y operaciones en memoria. Aun así, un diseño adecuado y consultas bien escritas favorecen enormemente el rendimiento general.

M. Seguridad de la base de datos

Importancia

La seguridad de los datos es un aspecto fundamental en cualquier sistema de gestión de información, especialmente cuando se manejan datos sensibles de personas como ocurre en el caso de la Fundación BisBlick. Es necesario garantizar que solo usuarios autorizados puedan acceder, modificar o eliminar los datos, y que ante fallos o ataques los datos estén protegidos y recuperables.

Estrategias de seguridad aplicadas o recomendadas

1. **Roles y permisos**

   En un entorno real de producción, se recomienda configurar diferentes roles en SQL Server para controlar el acceso a la base de datos. Algunos ejemplos aplicables:
   - Rol lectura (ROL_LECTURA):
     - Permisos: SELECT
     - Usuarios: Voluntarios que solo necesitan consultar datos
   - Rol edición (ROL_EDICION):
     - Permisos: SELECT, INSERT, UPDATE
     - Usuarios: Administradores de becas
   - Rol administrador (ROL_ADMIN):
     - Permisos: SELECT, INSERT, UPDATE, DELETE, CREATE PROCEDURE, etc.
     - Usuarios: Personal técnico o de sistemas

Ejemplo de creación de un rol con permisos de lectura

```
CREATE ROLE ROL_LECTURA;
GRANT SELECT ON SCHEMA :: dbo TO ROL_LECTURA;
```

```
EXEC sp_addrolemember 'ROL_LECTURA', 'usuario_lectura';
```

2. **Autenticación y control de acceso**
    - Se recomienda usar autenticación de SQL Server o Windows para validar usuarios.
    - Cada usuario debe tener su propia cuenta (evitar usuarios genéricos).
    - Se debe mantener un registro (auditoría) de accesos y modificaciones importantes.
3. **Backups y recuperación ante fallos**
    - Realizar copias de seguridad periódicas (diarias o semanales).
    - Guardar los backups en una ubicación segura y externa.
    - Documentar el procedimiento de restauración para actuar ante errores, fallos técnicos o pérdida de datos.
4. **Validación de datos a nivel SQL**
    - Se utilizaron restricciones como NOT NULL, UNIQUE, CHECK y claves foráneas (FK) para prevenir la carga de datos inválidos.
    - Se implementaron triggers y funciones para controlar ciertos cambios o automatizar validaciones, reforzando la seguridad lógica del sistema.
5. **Prevención de ataques (básica)**
    - Evitar uso de SQL dinámico sin parámetros en procedimientos (prevención de inyecciones SQL).
    - No exponer la base de datos directamente a usuarios sin intermediación de interfaces seguras.
    - Controlar longitudes máximas y tipos de datos para prevenir overflow o errores maliciosos.

O. Gestión de riesgos

La gestión de riesgos es una parte clave en todo proyecto tecnológico. Consiste en identificar posibles problemas que puedan afectar el desarrollo, la implementación o el uso futuro del sistema, y definir estrategias para reducir su probabilidad o impacto.

Durante el desarrollo del sistema de base de datos para la Fundación BisBlick se analizaron diversos factores técnicos, humanos y organizativos que podrían haber comprometido el éxito del proyecto.

Identificación de riesgos y estrategias de mitigación

A continuación, se detallan los principales riesgos identificados, su probabilidad e impacto estimado, junto con la estrategia adoptada para prevenirlos o corregirlos:

Riesgo	Probabilidad	Impacto	Estrategia de mitigación
Dificultad para interpretar correctamente los datos del Excel	Media	Alta	Realizar reuniones grupales de interpretación, validar campos con varios miembros del equipo.
Carga de datos inconsistentes o incompletos	Alta	Alta	Limpieza previa de datos en Excel, validación manual, uso de restricciones SQL.
Fallos en relaciones entre tablas	Media	Alta	Testeo de integridad referencial desde el inicio, uso de claves foráneas y pruebas de inserción.
Uso incorrecto de claves primarias o foráneas	Baja	Alta	Revisión cruzada del DER con los scripts antes de ejecutar cargas.
Errores de escritura en procedimientos y consultas	Media	Media	Uso del entorno SSMS con autocompletado, pruebas parciales por bloque.
Perdida de datos por errores en la transacción	Baja	Alta	Uso de bloques TRY/CATCH y transacciones explícitas (ROLLBACK y COMMIT).
Desorganización del equipo o tareas no claras	Media	Media	Asignación de roles (Product Owner, Developer, Documentador), planificación en Sprints.
Retrasos por carga académica u otros compromisos	Alta	Media	Distribución equilibrada de tareas, anticipación de partes críticas del proyecto.
Desalineación con los objetivos de la fundación	Baja	Alta	Revisión constante del propósito del sistema en relación con las necesidades de BisBlick.

P. Costos y presupuestos

Costos y presupuestos

Aunque el desarrollo del sistema fue realizado en un entorno académico, es posible estimar los costos que implicaría llevar a cabo este proyecto en un contexto real. Para ello se consideran los siguientes componentes:

1. **Recursos humanos**

   El mayor costo en un desarrollo de base de datos suele estar asociado a las horas de trabajo del equipo. Suponiendo un equipo de tres personas (analista funcional, desarrollador y tester/documentador), y un estimado de 100 horas totales distribuidas en cuatro sprints:

Rol	Cantidad	Horas estimadas	Costo por hora	Total
Analista funcional	1	30	$5.000	$150.000
Desarrollador SQL	1	40	$5.000	$200.000
Tester / Documentador	1	30	$4.000	$120.000
Subtotal				**$470.000**

2. **Hardware y Software**

Recurso	Detalle	Costo estimado
Equipos de desarrollo	PCs o notebooks (uso compartido)	$0 (ya disponibles)
Licencia SQL Server Developer	Versión gratuita para desarrollo	$0
Microsoft SQL Server (licencia)	Versión Standard (si se usara en producción)	$160.000 aprox.
Licencia sistema operativo	Windows Server (opcional)	$100.000 aprox.
Hosting / Infraestructura en nube	Servidor virtual básico (6 meses)	$120.000
Subtotal		**$380.000**

3. **Costo total estimado**

Categoría	**Costo**
Recursos humanos	$470.000
Hardware/Software	$380.000
Total estimado	**$850.000**

Consideraciones finales

Se utilizaron herramientas gratuitas o educativas (como SQL Server Developer y Excel) para evitar costos en el entorno de trabajo.
La estimación incluye infraestructura básica, pero en un proyecto real podría escalarse con backups automáticos, alta disponibilidad, etc.
Este análisis demuestra que, aun con costos razonables, se puede desarrollar un sistema profesional que aporta gran valor a organizaciones sociales como BisBlick.

Q. Conclusiones

El desarrollo del sistema de gestión de datos para la Fundación BisBlick permitió aplicar de manera práctica los conocimientos adquiridos sobre bases de datos relacionales, desde el diseño conceptual hasta la implementación física en SQL Server. A partir de una planilla real, se logró construir una solución estructurada, escalable y alineada con las necesidades reales de una organización social.

Entre los **principales hitos alcanzados** se destacan:

- La elaboración de un modelo entidad-relación (DER) detallado y normalizado.
- La implementación completa de una base de datos en SQL Server, con sus respectivas tablas, relaciones y restricciones de integridad.
- El desarrollo de consultas SQL optimizadas, procedimientos almacenados, vistas, triggers, funciones y una transacción controlada.
- La aplicación de índices y buenas prácticas de seguridad para mejorar el rendimiento y proteger los datos.
- El trabajo colaborativo con metodología SCRUM, con roles definidos y entregas parciales bien organizadas.

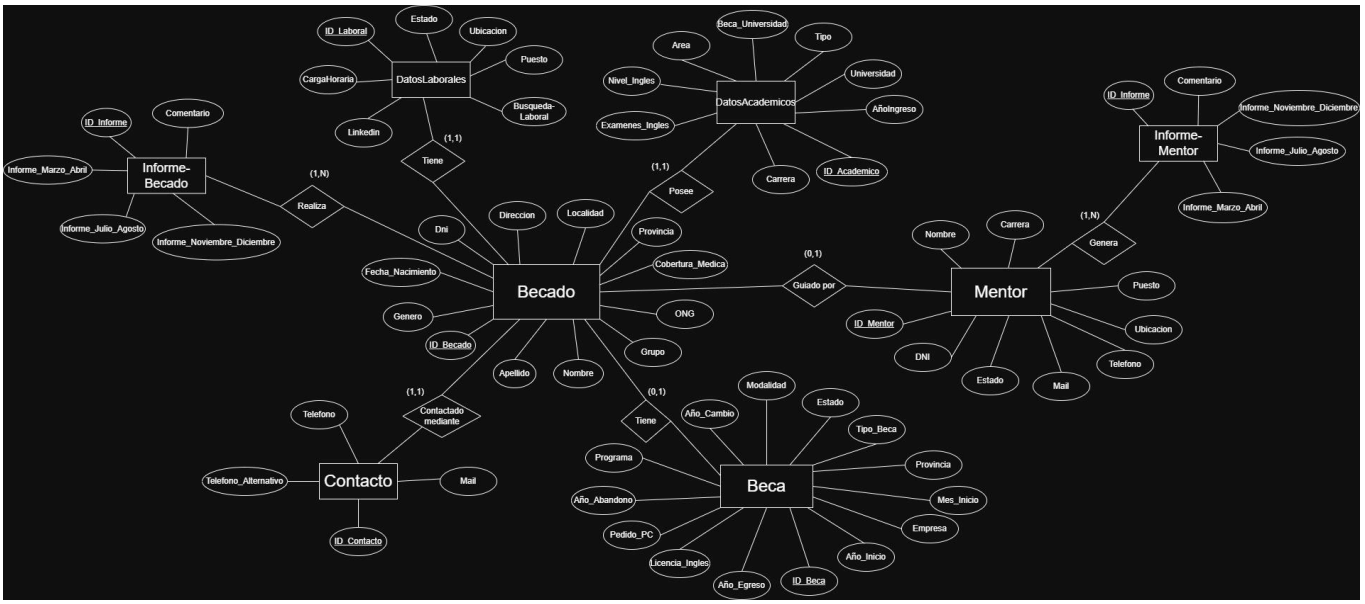
Entre los **desafíos más relevantes** que enfrentó el equipo se encuentran:

- La interpretación y limpieza de los datos originales en Excel, que presentaban inconsistencias y campos poco claros.
- El diseño de un modelo que representara fielmente la complejidad de las relaciones entre becados, programas, mentores e informes.
- La necesidad de asegurar la integridad y trazabilidad de los datos en todas las etapas del proyecto.

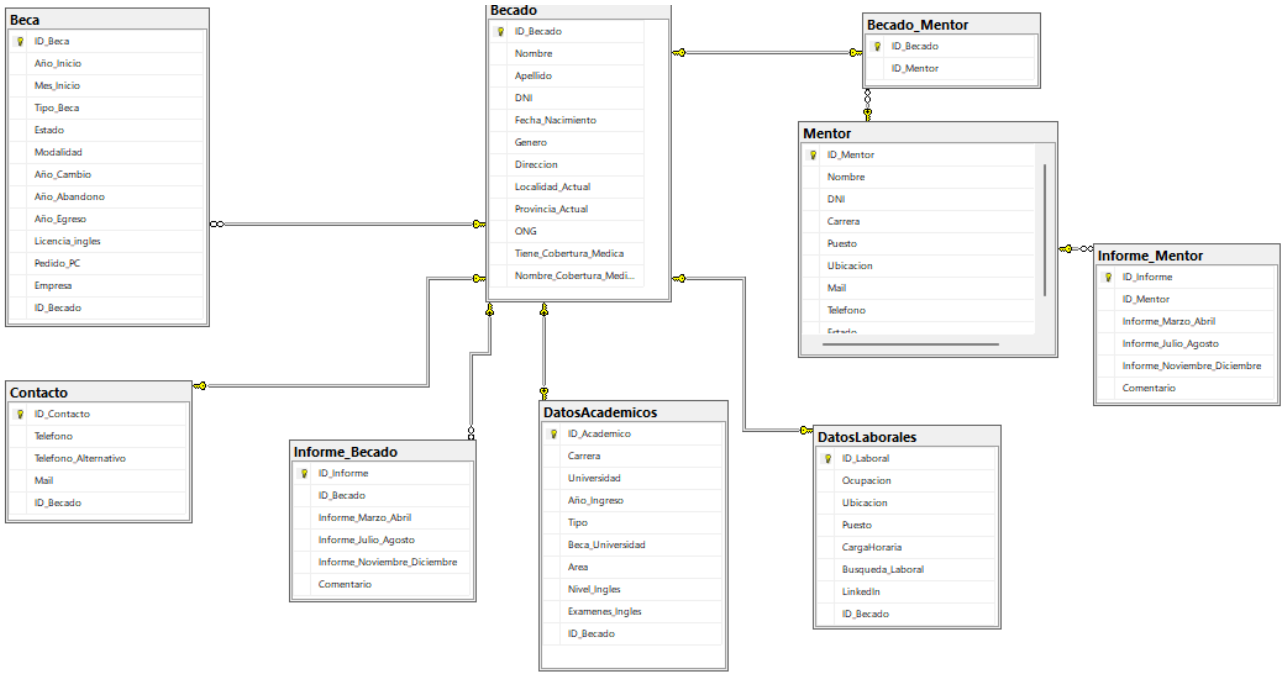
El proyecto fue exitoso tanto en términos académicos como prácticos. Se logró implementar una solución funcional, profesional y alineada con los valores de la Fundación BisBlick. Además, el equipo desarrolló habilidades técnicas, organizativas y de documentación que serán fundamentales en futuros desafíos del campo de los sistemas de información.

R. Anexos

Diagrama entidad-relación:



Modelado relacional:



S. Referencias

- Fundación BisBlick – Sitio oficial
  https://www.bisblick.org/
- Fundación BisBlick – Instagram oficial
  https://www.instagram.com/bisblick_talentojoven
- Microsoft SQL Server – Documentación oficial
  https://learn.microsoft.com/es-es/sql/sql-server/
- Diagrama Entidad-Relación (DER) – herramienta draw.io
  https://app.diagrams.net/
- Universidad Argentina de la Empresa (UADE) – Programa de estudios
  Contenidos de la asignatura Ingeniería de Datos I – 2025
- Excel/Hoja de cálculo “Seguimiento Camadas 2025 – Fundación BisBlick”
  Documento provisto por la fundación con datos reales de becados
- Apuntes personales y material de clase del docente Ing. Franco Emanuel Salazar