UNIVERSIDAD TECNOLÓGICA DE LA RIVIERA MAYA

TÉCNICO SUPERIOR UNIVERSITARIO EN DESARROLLO DE SOFTWARE MULTIPLATAFORMA

“PROYECTO INTEGRADORA”

PRESENTA:

DIEGO BLANCO RODRÍGUEZ

MOISES DAVID CASTAÑEDA MAY

ERICK DANIEL VARGAS CORONA

2402027

2402046

2402037

GENERACIÓN 2024-2028

EMPRESA: DREAM TEAM INTERNATIONAL SYSTEM ENTERTAINMENT

ASESOR EMPRESARIAL:

Grado abreviado. Nombre Apellidos

Playa del Carmen, Quintana Roo Septiembre

"Sprint 1: Diseño e Implementación de la Base de Datos"

Objetivo: Desarrollar la capacidad de analizar, diseñar e implementar una base de datos funcional que soporte los requerimientos del sistema definidos en el Sprint 0.

Descripción: Durante este Sprint 1, el equipo se ha centrado en la construcción de la base de datos del proyecto "TALENT BRIDGE". La actividad se ha dividido en tres fases clave:

1. Análisis de Requerimientos y Entidades.
El primer paso consistió en analizar a fondo los requerimientos funcionales y no funcionales (RF y RNF) del Sprint 0. A partir de este análisis, se identificaron todas las entidades principales (como Usuario, Estudiante, Empresa) y las entidades de relación (como Postulación, Mentoría) necesarias para dar soporte al sistema.

Para ver el desglose completo de las entidades identificadas, consulte el **Anexo 1.1-A-Tabla de requerimientos.**

2. Diseño Lógico y Físico de la Base de Datos.
Con las entidades definidas, se procedió a diseñar el esquema de la base de datos. Esta fase implicó la creación de un diseño lógico relacional, donde cada entidad se traduce en una tabla. Se definieron meticulosamente todas las columnas, sus tipos de datos, y las relaciones entre ellas, estableciendo claves primarias (PK) y foráneas (FK) para asegurar la integridad referencial.

Para ver el esquema detallado de tablas y sus relaciones, consulte el **Anexo-1.1-B-Modelo entidad-relación.**

3. Normalización de la Base de Datos (Hasta 3NF). Se aplicó un proceso de normalización para asegurar la integridad, eficiencia y evitar la redundancia de datos, cumpliendo hasta la Tercera Forma Normal (3NF):

- Primera Forma Normal (1NF): Se aseguró que todas las tablas contengan valores atómicos. No se almacenan listas o conjuntos de datos en una sola columna (por ejemplo, las habilidades de un estudiante no son un texto, sino registros en una tabla separada).
- Segunda Forma Normal (2NF): El diseño satisface la 2NF. Las tablas con claves primarias simples (como id_student en students) la cumplen automáticamente. En las nuevas tablas pivote con claves compuestas (como student_skills y offer_skills), no existen atributos no-clave que dependan parcialmente de la clave.
- Tercera Forma Normal (3NF): Se eliminaron las dependencias transitivas. Por ejemplo, para saber la escuela de un estudiante, se debe ir de students a careers y de careers a schools. El nombre de la escuela no se repite en la tabla students, evitando así que un atributo no-clave dependa de otro atributo no-clave.
- Resolución de M:M: Un punto crucial fue la resolución de relaciones Muchos a Muchos. Como un estudiante puede tener muchas habilidades y una habilidad puede pertenecer a *muchos* estudiantes, se creó la tabla pivote student_skills. De igual forma, se creó offer_skills para conectar offers y skills. Esto se refleja en el script de mejora v 1.1.sql.

4. Implementación en SGBD (PostgreSQL). El diseño físico se tradujo en código SQL (DDL) y se implementó exitosamente en el gestor de base de datos **PostgreSQL**. El script completo, que combina la creación inicial de tablas (primerpaso.sql) y las mejoras de normalización y funcionalidad (v 1.1.sql), se ha documentado. Para ver el código completo de la base de datos, consulte el **Anexo 1.1-C. Script SQL en PostgreSQL (DDL).**

5. Población de Datos (DML). Para validar el modelo y la integridad referencial, la base de datos se pobló con un conjunto de datos de prueba (DML). Se simularon registros para usuarios, estudiantes, empresas, ofertas, habilidades y las relaciones entre ellos. Para ver el script de inserción de datos, consulte el **Anexo 1.1-D- Script para insertar los datos (DML).**

6. Verificación de Integridad (DQL). Finalmente, se ejecutó un conjunto de consultas (DQL) para verificar que las relaciones (JOINs) y la integridad de los datos funcionan como se esperaba. Se probaron conexiones entre tablas primarias y pivote. Para ver las consultas de prueba, consulte el **Anexo 1.1-E- Consultas de Verificación (DQL)**

7. Gestión SCRUM. Todo el trabajo de este sprint fue gestionado siguiendo la metodología SCRUM. Las tareas fueron asignadas y su progreso documentado. Para ver los detalles de la planificación, consulte el **Anexo 1.1-F- Formato: Acta de Reunión**

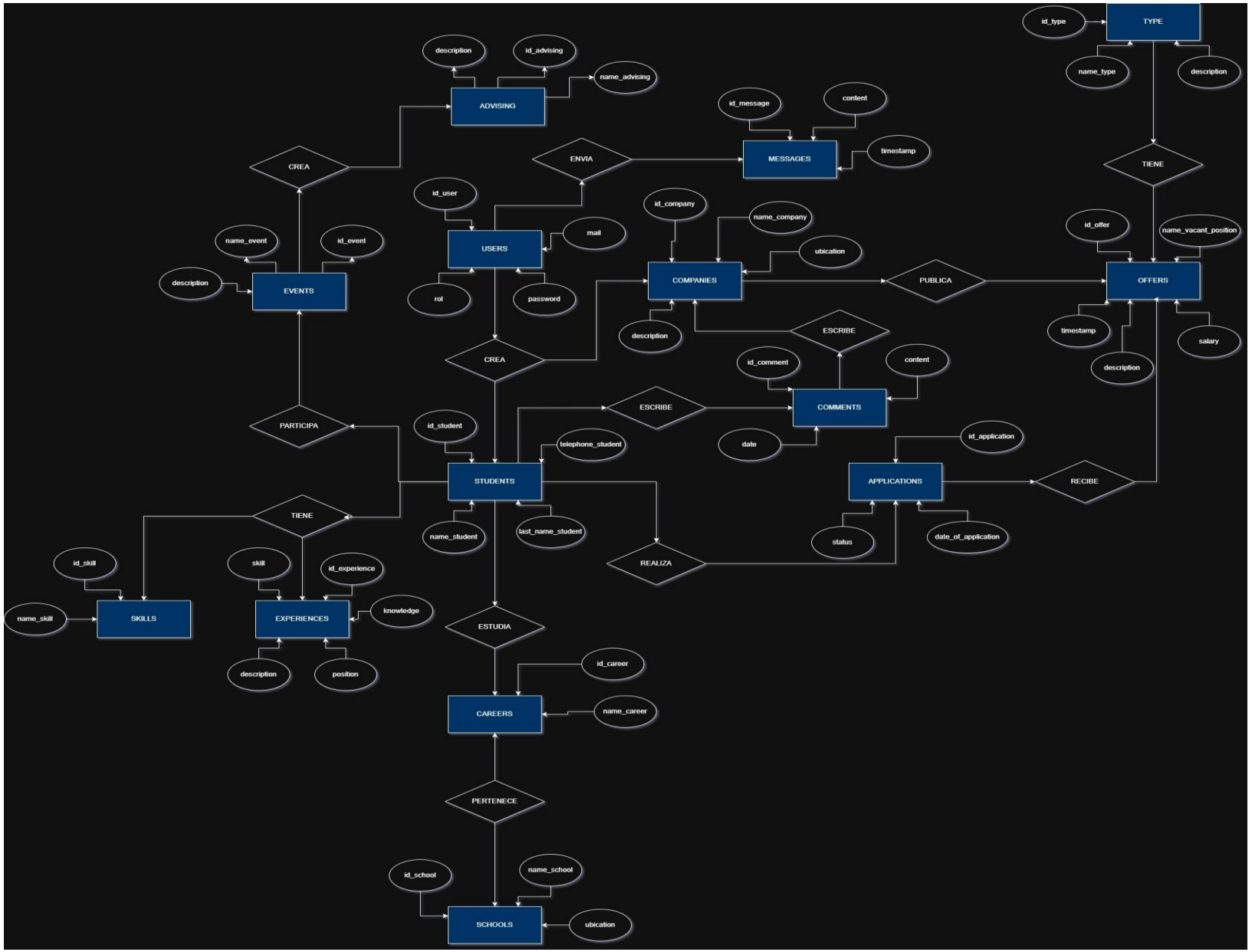
Anexos 1.1

A. Tabla de requerimientos.

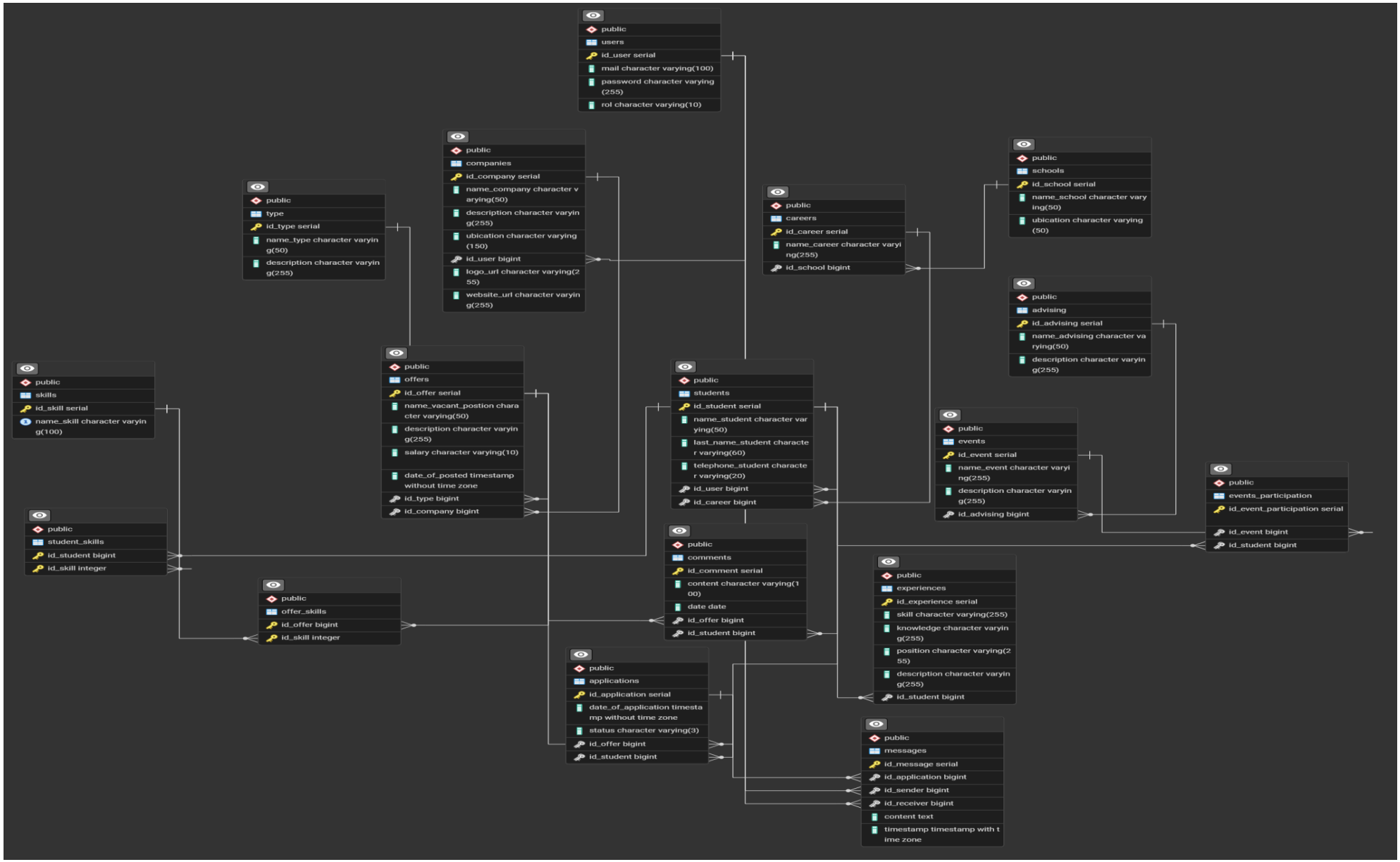
ID	Tipo de Requerimiento	Prioridad	Descripción del Requerimiento
RF-01	Funcional	Imprescindible	El sistema debe permitir el registro de nuevos usuarios con dos roles distintos: "Estudiante" y "Empleador".
RF-02	Funcional	Imprescindible	El sistema debe validar las credenciales de los usuarios para permitir el inicio de sesión (autenticación).
RF-03	Funcional	Imprescindible	Los usuarios "Estudiante" deben poder crear y editar un perfil profesional (datos de contacto, habilidades, formación, etc.).
RF-04	Funcional	Imprescindible	Los usuarios "Empleador" deben poder crear y editar un perfil de su empresa.
RF-05	Funcional	Imprescindible	Los usuarios "Empleador" deben poder publicar, editar y eliminar ofertas de empleo.
RF-06	Funcional	Imprescindible	Los usuarios "Estudiante" deben poder buscar y filtrar ofertas de empleo (por área, tipo de contrato, etc.).
RF-07	Funcional	Deseable	El sistema debe permitir a los estudiantes postularse a las ofertas de empleo directamente desde la plataforma.
RF-08	Funcional	Deseable	Debe existir un sistema de mensajería interna para que empleadores y estudiantes puedan comunicarse.

ID	Tipo de Requerimiento	Prioridad	Descripción del Requerimiento
RF-09	Funcional	Deseable	Se debe incluir un módulo de mentorías donde profesionales puedan ofrecer guía a los estudiantes.
RF-10	Funcional	Opcional	La plataforma podría contar con una sección de recursos con artículos y consejos sobre desarrollo profesional.
RF-11	Funcional	Opcional	Implementar un sistema de notificaciones por correo electrónico para nuevas ofertas o postulaciones.
RNF-01	No Funcional	Imprescindible	**Seguridad:** La información personal de los usuarios debe estar protegida y encriptada. El acceso debe estar restringido por rol.
RNF-02	No Funcional	Imprescindible	**Usabilidad:** La interfaz de la plataforma debe ser intuitiva, clara y fácil de navegar para ambos tipos de usuario.
RNF-03	No Funcional	Deseable	**Rendimiento:** El tiempo de respuesta del sistema ante cualquier acción del usuario no debe superar los 2 segundos.
RNF-04	No Funcional	Deseable	**Compatibilidad:** La aplicación web debe ser compatible con las últimas versiones de los navegadores más populares.

B.Modelo entidad-relación.(Actualizado)



Modelo relacional. (Actualizado.)



C. Script SQL en PostgreSQL (DDL).

```
-- This script was generated by the ERD tool in pgAdmin 4.
BEGIN;
-- ------------------------------------------------------------------
-- SECCIÓN 1: CREACIÓN DE TABLAS BASE (primerpaso.sql)
-- ------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS public.users
( id_user serial NOT NULL,
    mail character varying(100) COLLATE pg_catalog."default" NOT NULL UNIQUE,
    password character varying(255) COLLATE pg_catalog."default" NOT NULL,
    -- Se añade el rol para diferenciar usuarios
    role character varying(10) DEFAULT 'student'::character varying NOT NULL,
    CONSTRAINT users_pkey PRIMARY KEY (id_user)
);

CREATE TABLE IF NOT EXISTS public.schools
(  id_school serial NOT NULL,
    name_school character varying(50) COLLATE pg_catalog."default" NOT NULL,
    ubication character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT schools_pkey PRIMARY KEY (id_school)
);

CREATE TABLE IF NOT EXISTS public.careers
( id_career serial NOT NULL,
    name_career character varying(255) COLLATE pg_catalog."default" NOT NULL,
    id_school bigint NOT NULL,
    CONSTRAINT careers_pkey PRIMARY KEY (id_career),
```

```
    FOREIGN KEY (id_school) REFERENCES public.schools (id_school)
);

CREATE TABLE IF NOT EXISTS public.students
(id_student serial NOT NULL,
    name_student character varying(50) COLLATE pg_catalog."default" NOT NULL,
    last_name_student character varying(60) COLLATE pg_catalog."default" NOT NULL,
    telephone_student character varying(20) COLLATE pg_catalog."default" NOT NULL,
    id_user bigint NOT NULL,
    id_career bigint NOT NULL,
    CONSTRAINT students_pkey PRIMARY KEY (id_student),
    FOREIGN KEY (id_user) REFERENCES public.users (id_user) ON DELETE
CASCADE,
    FOREIGN KEY (id_career) REFERENCES public.careers (id_career)
);

CREATE TABLE IF NOT EXISTS public.companies
(id_company serial NOT NULL,
    name_company character varying(50) COLLATE pg_catalog."default" NOT NULL,
    description character varying(255) COLLATE pg_catalog."default" NOT NULL,
    ubication character varying(150) COLLATE pg_catalog."default" NOT NULL,
    id_user bigint NOT NULL,
    CONSTRAINT companies_pkey PRIMARY KEY (id_company),
    FOREIGN KEY (id_user) REFERENCES public.users (id_user) ON DELETE
CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS public.type
(id_type serial NOT NULL,
    name_type character varying(50) COLLATE pg_catalog."default" NOT NULL,
    description character varying(255) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT type_pkey PRIMARY KEY (id_type)
);

CREATE TABLE IF NOT EXISTS public.offers
(id_offer serial NOT NULL,
    name_vacant_postion character varying(50) COLLATE pg_catalog."default" NOT
NULL,
    description character varying(255) COLLATE pg_catalog."default" NOT NULL,
    salary character varying(10) COLLATE pg_catalog."default" NOT NULL,
    date_of_posted timestamp without time zone NOT NULL,
    id_type bigint NOT NULL,
    id_company bigint NOT NULL,
    CONSTRAINT offers_pkey PRIMARY KEY (id_offer),
    FOREIGN KEY (id_type) REFERENCES public.type (id_type),
    FOREIGN KEY (id_company) REFERENCES public.companies (id_company) ON
DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS public.experiences
(id_experience serial NOT NULL,
    skill character varying(255) COLLATE pg_catalog."default" NOT NULL,
    knowledge character varying(255) COLLATE pg_catalog."default" NOT NULL,
    "position" character varying(255) COLLATE pg_catalog."default" NOT NULL,
    description character varying(255) COLLATE pg_catalog."default" NOT NULL,
    id_student bigint NOT NULL,
```

```
    CONSTRAINT experiences_pkey PRIMARY KEY (id_experience),
    FOREIGN KEY (id_student) REFERENCES public.students (id_student) ON DELETE
CASCADE
);

CREATE TABLE IF NOT EXISTS public.applications
(id_application serial NOT NULL,
    date_of_application timestamp without time zone NOT NULL,
    status character varying(3) COLLATE pg_catalog."default" NOT NULL,
    id_offer bigint NOT NULL,
    id_student bigint NOT NULL,
    CONSTRAINT applications_pkey PRIMARY KEY (id_application),
    FOREIGN KEY (id_offer) REFERENCES public.offers (id_offer) ON DELETE
CASCADE,
    FOREIGN KEY (id_student) REFERENCES public.students (id_student) ON DELETE
CASCADE
);

CREATE TABLE IF NOT EXISTS public.comments
(id_comment serial NOT NULL,
    content character varying(100) COLLATE pg_catalog."default" NOT NULL,
    date date NOT NULL,
    id_offer bigint NOT NULL,
    id_student bigint NOT NULL,
    CONSTRAINT comments_pkey PRIMARY KEY (id_comment),
    FOREIGN KEY (id_offer) REFERENCES public.offers (id_offer),
    FOREIGN KEY (id_student) REFERENCES public.students (id_student)
);
```

```
CREATE TABLE IF NOT EXISTS public.advising
(id_advising serial NOT NULL,
    name_advising character varying(50) COLLATE pg_catalog."default" NOT NULL,
    description character varying(255) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT advising_pkey PRIMARY KEY (id_advising)
);

CREATE TABLE IF NOT EXISTS public.events
(id_event serial NOT NULL,
    name_event character varying(255) COLLATE pg_catalog."default" NOT NULL,
    description character varying(255) COLLATE pg_catalog."default" NOT NULL,
    id_advising bigint NOT NULL,
    CONSTRAINT events_pkey PRIMARY KEY (id_event),
    FOREIGN KEY (id_advising) REFERENCES public.advising (id_advising)
);

CREATE TABLE IF NOT EXISTS public.events_participation
(id_event_participation serial NOT NULL,
    id_event bigint NOT NULL,
    id_student bigint NOT NULL,
    CONSTRAINT events_participation_pkey PRIMARY KEY (id_event_participation),
    FOREIGN KEY (id_event) REFERENCES public.events (id_event),
    FOREIGN KEY (id_student) REFERENCES public.students (id_student)
);
```

```
-- ------------------------------------------------------------------
-- SECCIÓN 2: MEJORAS AL PERFIL DE 'companies' (v 1.1.sql)
-- ------------------------------------------------------------------

ALTER TABLE companies
ADD COLUMN IF NOT EXISTS logo_url VARCHAR(255),
ADD COLUMN IF NOT EXISTS website_url VARCHAR(255);
-- ------------------------------------------------------------------
-- SECCIÓN 3: CREACIÓN DEL SISTEMA DE "SKILLS" (v 1.1.sql)
-- (Normalización para eliminar M:M)
-- ------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS public.skills
(id_skill SERIAL PRIMARY KEY,
    name_skill VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS public.student_skills
(id_student BIGINT NOT NULL,
    id_skill INT NOT NULL,
    PRIMARY KEY (id_student, id_skill), -- Clave primaria compuesta
    FOREIGN KEY (id_student) REFERENCES students(id_student) ON DELETE
CASCADE,
    FOREIGN KEY (id_skill) REFERENCES skills(id_skill) ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS public.offer_skills (
    id_offer BIGINT NOT NULL,
    id_skill INT NOT NULL,
    PRIMARY KEY (id_offer, id_skill), -- Clave primaria compuesta
    FOREIGN KEY (id_offer) REFERENCES offers(id_offer) ON DELETE CASCADE,
    FOREIGN KEY (id_skill) REFERENCES skills(id_skill) ON DELETE CASCADE
);
-- ------------------------------------------------------------------
-- SECCIÓN 4: CREACIÓN DE TABLA 'messages' (v 1.1.sql)
-- ------------------------------------------------------------------
CREATE TABLE IF NOT EXISTS public.messages
 (id_message SERIAL PRIMARY KEY,
    id_application BIGINT NOT NULL, -- A qué postulación pertenece este chat
    id_sender BIGINT NOT NULL,      -- Quién envía el mensaje (FK a users)
    id_receiver BIGINT NOT NULL,    -- Quién recibe el mensaje (FK a users)
    content TEXT NOT NULL,
    "timestamp" TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_application) REFERENCES applications(id_application) ON
DELETE CASCADE,
    FOREIGN KEY (id_sender) REFERENCES users(id_user),
    FOREIGN KEY (id_receiver) REFERENCES users(id_user)
);


END;
```

D- Script para insertar los datos (DML).

```
BEGIN;

-- 1. Insertar roles básicos

INSERT INTO public.schools (name_school, ubication) VALUES

('Universidad Tecnológica', 'Playa del Carmen');


INSERT INTO public.careers (name_career, id_school) VALUES

('TSU en Desarrollo de Software', 1);


INSERT INTO public.type (name_type, description) VALUES

('Medio Tiempo', 'Trabajos de 4-6 horas diarias'),

('Tiempo Completo', 'Trabajos de 8 horas diarias');


-- 2. Insertar Usuarios (1 Estudiante, 1 Empleador)

-- Contraseñas son 'pass123' (hasheadas en un sistema real)

INSERT INTO public.users (mail, password, role) VALUES

('diego@utcancun.edu.mx', 'pass123', 'student'),

('erick@empresa.com', 'pass123', 'company');


-- 3. Insertar Perfiles (Estudiante y Empresa)

INSERT INTO public.students (name_student, last_name_student,telephone_student,
id_user, id_career) VALUES

('Diego', 'Blanco','9841234567', 1, 1);


INSERT INTO public.companies (name_company, description, ubication, id_user,
logo_url, website_url) VALUES

('Empresa XYZ Tech', 'Desarrollo de software y soluciones', 'Cancún', 2,
'http://logo.com/img.png', 'http://empresa.com');
```

```
-- 4. Insertar Habilidades (Skills)

INSERT INTO public.skills (name_skill) VALUES ('React'), ('Node.js'), ('PostgreSQL'),
('Ventas');


-- 5. Insertar Ofertas de Trabajo

INSERT INTO public.offers (name_vacant_postion, description, salary, date_of_posted,
id_type, id_company) VALUES

('Desarrollador Frontend Jr', 'Buscamos dev con ganas de aprender React', '10000', NOW(),
1, 1),

('Desarrollador Backend Jr', 'Buscamos dev con ganas de aprender Node.js', '12000',
NOW(), 1, 1);


-- 6. Conectar Habilidades (Tablas Pivote)

-- Asignar skills a Diego (React y PostgreSQL)

INSERT INTO public.student_skills (id_student, id_skill) VALUES

(1, 1), -- React

(1, 3); -- PostgreSQL


-- Asignar skills requeridas a la oferta 'Frontend Jr' (React)

INSERT INTO public.offer_skills (id_offer, id_skill) VALUES

(1, 1); -- React


-- Asignar skills requeridas a la oferta 'Backend Jr' (Node.js y PostgreSQL)

INSERT INTO public.offer_skills (id_offer, id_skill) VALUES

(2, 2), -- Node.js

(2, 3); -- PostgreSQL
```

-- 7. Crear una postulación

INSERT INTO public.applications (date_of_application, status, id_offer, id_student)
VALUES

(NOW(), 'En', 1, 1); -- Diego se postula a Frontend Jr

COMMIT;

E- Consultas de Verificación (DQL)

-- Consulta 1: Ver el perfil de un estudiante y su carrera

-- (Prueba: JOIN entre students, careers y schools)

```
SELECT
    s.name_student,
    s.last_name_student,
    c.name_career,
    sc.name_school
FROM
    public.students s
JOIN
    public.careers c ON s.id_career = c.id_career
JOIN
    public.schools sc ON c.id_school = sc.id_school
WHERE
    s.id_student = 1;
```

	name_student character varying (50)	last_name_student character varying (60)	name_career character varying (255)	name_school character varying (50)
1	Diego	Blanco	TSU en Desarrollo de Softw...	Universidad Tecnológi...

-- Resultado Esperado: 'Diego', 'Blanco', 'TSU en Desarrollo de Software', 'Universidad Tecnológica'

-- ----------

-- Consulta 2: Ver las ofertas publicadas por una compañía

-- (Prueba: JOIN entre companies y offers)

```
SELECT
    c.name_company,
    o.name_vacant_postion,
    o.salary
FROM
    public.offers o
JOIN
    public.companies c ON o.id_company = c.id_company
WHERE
    c.id_company = 1;
```

	name_company character varying (50)	name_vacant_postion character varying (50)	salary character varying (10)
1	Empresa XYZ Tech	Desarrollador Frontend...	10000
2	Empresa XYZ Tech	Desarrollador Backend ...	12000

-- Resultado Esperado: 'Empresa XYZ Tech', 'Desarrollador Frontend Jr', '10000'

-- 'Empresa XYZ Tech', 'Desarrollador Backend Jr', '12000'

-- ----------

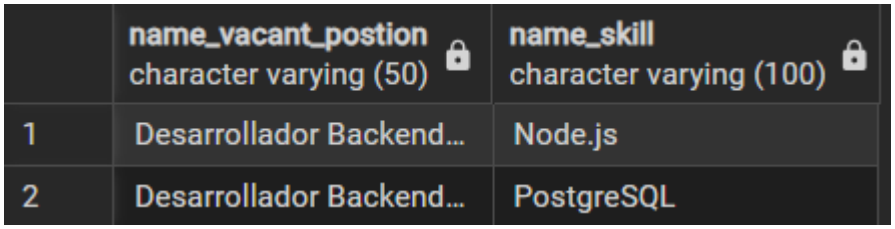
```
-- Consulta 3: Ver qué estudiantes tienen la habilidad "React"
-- (Prueba: JOIN M:M entre students, student_skills y skills)
SELECT
    s.name_student,
    sk.name_skill
FROM
    public.students s
JOIN
    public.student_skills ss ON s.id_student = ss.id_student
JOIN
    public.skills sk ON ss.id_skill = sk.id_skill
WHERE
    sk.name_skill = 'React';
```

	name_student character varying (50)	name_skill character varying (100)
1	Diego	React

```
-- Resultado Esperado: 'Diego', 'React'

-- ------------------------------------------------------------
```

```
-- Consulta 4: Ver qué habilidades se requieren para la oferta "Backend Jr"
-- (Prueba: JOIN M:M entre offers, offer_skills y skills)
SELECT
    o.name_vacant_postion,
    sk.name_skill
FROM
    public.offers o
JOIN
    public.offer_skills os ON o.id_offer = os.id_offer
JOIN
    public.skills sk ON os.id_skill = sk.id_skill
WHERE
    o.name_vacant_postion = 'Desarrollador Backend Jr';
```



	name_vacant_postion character varying (50)	name_skill character varying (100)
1	Desarrollador Backend...	Node.js
2	Desarrollador Backend...	PostgreSQL

```
-- Resultado Esperado: 'Desarrollador Backend Jr', 'Node.js'
--                     'Desarrollador Backend Jr', 'PostgreSQL'

-- ------------------------------------------------------------
```

```
-- Consulta 5: Ver las postulaciones de un estudiante
-- (Prueba: JOIN entre applications, students y offers)
SELECT
    s.name_student,
    o.name_vacant_postion,
    a.status
FROM
    public.applications a
JOIN
    public.students s ON a.id_student = s.id_student
JOIN
    public.offers o ON a.id_offer = o.id_offer
WHERE
    s.id_student = 1;
```

	name_student character varying (50)	name_vacant_postion character varying (50)	status character varying (3)
1	Diego	Desarrollador Frontend...	En

```
-- Resultado Esperado: 'Diego', 'Desarrollador Frontend Jr', 'En'
--------------------------------------------------------------------------------
```

F- Formato: Acta de Reunión

1.Datos Generales

• Proyecto: TALENT BRIDGE

• Fecha: _17 de octubre del 2025____

• Hora de inicio: _11:10 a.m Hora de término: ___12:00_p.m___

• Lugar / Medio (Presencial / Virtual): __Presencial_

• Tipo de reunión (Planificación / Seguimiento / Revisión / Cierre): _ __Planificación____

• Responsable de la reunión: _Diego Blanco Rodriguez____

• Elaboró el acta: __Diego Blanco Rodriguez_____

2. Participantes

Nombre	*Cargo / Rol*	*Asistencia (Sí/No)*
Diego Blanco Rodriguez	Scrum Master/ Desarrollador	Sí
Erick Corona Vargas	Product Owner/Desarrollador	Si
Moises David Castañeda May	Desarrollador	Si

3. Orden del Día

N.º	**TEMA / PUNTO A TRATAR**
1	Revisión del planteamiento del problema.
2	Definición de objetivos generales y específicos.
3	Estructura de la plataforma digital.
4	Asignación de tareas iniciales.
5	Definición de conceptos y análisis de entidades.
6	Realización del modelo entidad-relación (DER) y relacional.
7	Realización de la base de datos funcional.
8	Consulta de la base de datos.

4. Desarrollo de la Reunión

Se discutió la problemática de empleabilidad juvenil en Playa del Carmen, se revisaron los objetivos del proyecto y se acordó diseñar una plataforma que conecte estudiantes con empleadores. Se asignaron tareas para el desarrollo del prototipo inicial, definiendo que el **Sprint 1 se centraría en el diseño e implementación de la base de datos** necesaria para la estructura de la plataforma.

5. Acuerdos y Compromisos

N.º	*Acuerdo / Compromiso*	*Responsable*	*Fecha Compromiso*	*Estado*
1	Diseñar el prototipo de la plataforma	Desarrolladores	15/10/2025	Completado
2	Desarrollar el módulo de autenticación	Scrum master	16/10/2025	Completado
3	Redactar el marco teorico	Product Owner	14/10/2025	Completado
4	Análisis de entidades de la BD	Equipo de desarrollo	17/10/2025	Completado
5	Diseño lógico y físico de la BD	Equipo de desarrollo	17/10/2025	Completado
6	Implementación y script SGBD	Equipo de desarrollo	02/11/2025	Completado
7	Pruebas de integridad y consultas	Equipo de desarrollo	02/11/2025	Completado

6. Observaciones Generales

Se sugirió incluir un módulo de mentorías empresariales y filtros de inclusión para estudiantes en situación vulnerable.

Actualización (Post-Sprint 1): El diseño e implementación de la base de datos (Sprint 1) se completó exitosamente. La estructura de la BD ya incluye las tablas necesarias para soportar los requerimientos funcionales (RF-01 a RF-07) y está preparada para los módulos deseados de mentorías (RF-09) y recursos (RF-10).

7. Cierre de la Reunión

Hora de cierre: _12:00 p.m_

Próxima reunión: _24 de octubre del 2025 11:10 a.m___ (Fecha y hora)

8. Firmas de Conformidad

Scrum Master: _Diego Blanco Rodriguez_____ Firma: _____

Product Owner: _Erick Corona Vargas_____ Firma: _____

Representante del Equipo: Diego Blanco Rodriguez___ Firma: _____