




**Universidad  
del Valle**

---


Proyecto Bases de Datos 'Mande'  
Informe de Resultados y Cierre de Proyecto

Versión: 001  
Fecha: 12/08/2022


	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

## HOJA DE CONTROL

<b>Institución</b>	Universidad del Valle
<b>Proyecto</b>	Proyecto Bases de Datos 'Mande'
<b>Entregable a</b>	Andres M. Castillo
<b>Autor(es)</b>	Carolain Jimenez Bedoya, Natalia Lopez Osorio, Hernando Lopez Rincon
<b>Fecha</b>	12 de Agosto de 2022

	<p>Proyecto Bases de Datos ‘Mande’ Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	---	------------------------------

<b>INTRODUCCIÓN</b>	<b>4</b>
1.1. Objeto	4
1.2. Alcance	4
<b>DESCRIPCIÓN DEL PROYECTO</b>	<b>4</b>
<b>DESARROLLO DEL PROYECTO</b>	<b>5</b>
3.1. FrontEnd	5
3.2. Creación base de datos	6
<b>EVALUACIÓN DEL PROYECTO</b>	<b>20</b>
4.1. Aspectos positivos	20
4.2. Aspectos negativos	22

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

## 1. INTRODUCCIÓN

### 1.1. Objeto

El objetivo de este informe es presentar el desarrollo del proyecto de Bases de datos para la aplicación Mande, aplicando los conocimientos adquiridos en el curso de los Modelos Entidad-Relación y Relacional, así como SQL, relación cliente-servidor y contenedores.

### 1.2. Alcance


El alcance final de este proyecto parte de una oferta-demanda de servicios básicos como pintores, mecánicos, jardineros, etc. Usuarios clientes que demandan dichos servicios y usuarios trabajadores que ofertan los servicios. Dichos usuarios pueden ser cualquier persona con acceso a internet y que se registre como cualquier tipo de usuario. Cabe destacar, que para realizar un registro de una misma persona como cliente o trabajador, estos deben suministrar correo y números de teléfono distintos para el correcto funcionamiento de la plataforma.

## 2. DESCRIPCIÓN DEL PROYECTO

El proyecto ha sido desarrollado con el objetivo de proveer soluciones a problemas diarios, al alcance de todos. En el mismo, la plataforma está diseñada para el uso de dos usuarios: trabajador y cliente.

En perfil de trabajador se pueden encontrar las siguientes acciones realizables dentro del aplicativo:

- Debe registrarse en el formulario de registro brindando por la plataforma.
- Puede iniciar sesión teniendo en cuenta su correo y contraseña.
- El trabajador solo puede ingresar una labor en el registro con su respectivo precio y unidad, sin embargo, dentro de la plataforma puede añadir más labores.
- Si el trabajador desea registrarse como cliente, este debe proporcionar un número de teléfono y correo electrónico distinto.
- El trabajador puede editar su información dentro de la plataforma.
- Puede descargar reportes de cobros y de servicios realizados, es decir, la plataforma cuenta con un historial.
- Puede verificar las reseñas y puntuaciones suministradas por los clientes a los que les ha realizado un trabajo.
- Puede salir de la plataforma cuando lo desee.

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

En perfil usuario se pueden encontrar las siguientes acciones:

- Debe registrarse en el formulario de registro brindando por la plataforma.
- Puede iniciar sesión teniendo en cuenta su correo y contraseña.
- Puede solicitar servicios a partir de los trabajadores que se encuentran disponibles.
- Puede proporcionar un método de pago.
- Puede puntuar y dar reseñas sobre los servicios adquiridos y concluidos.
- Envía solicitudes de trabajo a los trabajadores.

### 3. DESARROLLO DEL PROYECTO

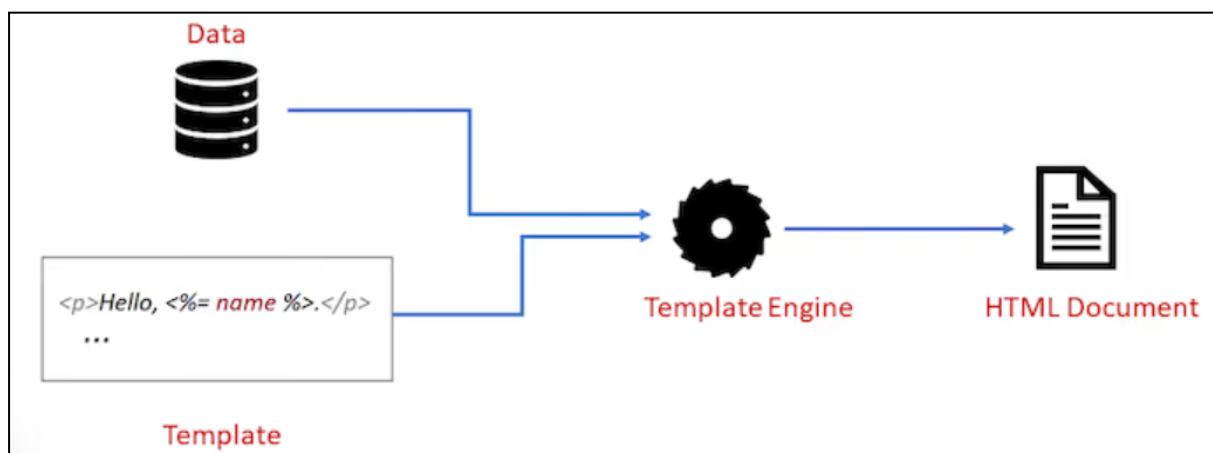
Para el desarrollo del proyecto se usó PostgreSQL, y .EJS y .JS.

PostgreSQL es el sistema gestor de base de datos de uso libre con el cual se implementó la creación de las tablas para la base de datos donde se define la tabla usuario, trabajador y cliente.


#### 3.1. FrontEnd

Para el FrontEnd se usó Embedded Javascript(.EJS), el cual es un sistema de plantillas en donde se define las páginas HTML en la sintaxis de EJS y especifica dónde irán los distintos datos en la página. Luego, la aplicación combina los datos con la plantilla y "renderiza" una página HTML completa en la que EJS toma tus datos y los inserta en la página web de acuerdo a cómo hayas definido la plantilla.

Las plantillas se encargan de la tarea de interpolar los datos en el código HTML.



Para el BackEnd, se implementó el uso de Javascript

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

### 3.2. Creación base de datos

Así se construyó la base de datos en PostgreSQL:

1. Se crea la base de datos con la siguiente estructura:

```
CREATE DATABASE mande_db
WITH
OWNER = postgres
ENCODING = 'UTF8'
LC_COLLATE = 'C'
LC_CTYPE = 'C'
TABLESPACE = pg_default
CONNECTION LIMIT = -1|
TEMPLATE template0;
```

2. Ya creada la base de datos, se inicia la tabla 'usuario':

```
CREATE TABLE USUARIO (
id_usuario SERIAL NOT NULL PRIMARY KEY,
nombre VARCHAR(64),
apellido VARCHAR(64),
email VARCHAR(64) UNIQUE NOT NULL,
telefono INTEGER UNIQUE NOT NULL,
cedula INTEGER,
contraseña VARCHAR(15),
direccion_nombre VARCHAR(64),
direccion POINT,
ciudad VARCHAR(64)
);
```

3. Las siguientes tablas en ser creadas, serán las de 'cliente' y 'trabajador':



```
CREATE TABLE TRABAJADOR (  
  id_trabajador SERIAL PRIMARY KEY,  
  id_usuario INT NOT NULL UNIQUE,  
  foto_perfil BYTEA,  
  estado_trabajador BOOL,  
  cuenta_trabajador INT,  
  
  FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)  
  
);
```

```
CREATE TYPE tipo_pago AS ENUM ('credito', 'debito');  
  
CREATE TABLE CLIENTE(  
  id_cliente SERIAL PRIMARY KEY,  
  id_usuario INT NOT NULL UNIQUE,  
  recibo BYTEA,  
  medio_pago tipo_pago,  
  
  FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)  
  
);
```

4. Una vez hecho esto, se crea la tabla 'labor' que almacena todas las labores que el 'cliente' puede solicitar, y el 'trabajador' realizar.

```
CREATE TABLE LABOR(  
  id_labor SERIAL PRIMARY KEY,  
  nombre_labor VARCHAR(64)  
  
);
```



5. Luego, creamos la tabla 'trabajador\_labor', que relaciona las labores que tienen registradas los trabajadores.

```
CREATE TYPE unidad AS ENUM ('dia', 'hora');

CREATE TABLE TRABAJADOR_LABOR(
  id_trabajador_labor SERIAL PRIMARY KEY,
  id_trabajador INT NOT NULL,
  id_labor INT NOT NULL,
  precio_labor INTEGER,
  unidad_valor unidad,

  FOREIGN KEY (id_trabajador) REFERENCES trabajador(id_trabajador),
  FOREIGN KEY (id_labor) REFERENCES labor(id_labor)
);
```

6. Luego creamos la tabla 'servicio', donde se va a almacenar todas las labores realizadas por 'trabajador' al 'cliente', incluye los datos de la fecha del servicio

```
CREATE TABLE SERVICIO(
  id_servicio SERIAL PRIMARY KEY,
  id_cliente INT NOT NULL,
  id_trabajador_labor INT NOT NULL,
  fecha_inicio DATE,
  fecha_final DATE,
  estado_servicio BOOL,

  FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente),
  FOREIGN KEY (id_trabajador_labor) REFERENCES trabajador_labor(id_trabajador_labor)
);
```

7. Por último, se tiene la tabla 'pago\_servicio', la cual captura la información necesaria para registrar los pagos generados por cada servicio.





```
CREATE TABLE PAGO_SERVICIO(  
id_pago SERIAL PRIMARY KEY,  
id_servicio INT NOT NULL UNIQUE,  
fecha_pago DATE,  
puntaje_trabajador FLOAT,  
reseña VARCHAR(100) DEFAULT ' ',  
  
FOREIGN KEY (id_servicio) REFERENCES servicio(id_servicio)  
);
```

Una vez creadas todas las tablas, se procede a insertar las labores que estarán disponibles para los trabajadores.

```
INSERT INTO labor(nombre_labor) VALUES('Mecanico');  
INSERT INTO labor(nombre_labor) VALUES ('Carpintero');  
INSERT INTO labor(nombre_labor) VALUES ('Albañil');  
INSERT INTO labor(nombre_labor) VALUES ('Pintor');  
INSERT INTO labor(nombre_labor) VALUES ('Cerrajero');  
INSERT INTO labor(nombre_labor) VALUES ('Peluquero');  
INSERT INTO labor(nombre_labor) VALUES ('Paseador de perros');  
INSERT INTO labor(nombre_labor) VALUES ('Cocinero');  
INSERT INTO labor(nombre_labor) VALUES ('Niño');  
INSERT INTO labor(nombre_labor) VALUES ('Tutor');  
INSERT INTO labor(nombre_labor) VALUES ('Chofer');  
INSERT INTO labor(nombre_labor) VALUES ('Exterminador');
```

A partir de este punto, se crean funciones de validación de registro de cliente y trabajador, y login.

Para la validación del registro del trabajador se implementó la siguiente función:



Proyecto Bases de Datos 'Mande'  
Informe de Resultados y Cierre de Proyecto

Universidad del Valle

---


```
Create or replace Function validar_registro_trabajador( nombre VARCHAR(64), apellido
VARCHAR(64), email_usuario VARCHAR(64),telefono INTEGER, cedula INTEGER, contraseña
VARCHAR(15), direccion_nombre VARCHAR(64), direccion POINT, ciudad VARCHAR(64),
foto_perfil BYTEA, cuenta_trabajador INTEGER, id_labor_t INTEGER, precio INT, und unidad )
Returns INTEGER
As $$
DECLARE
id integer;
id_trabajador_t INTEGER;

Begin
  If NOT Exists (
    Select
      *
    From
      usuario
    Where
      -- Assuming all three fields are primary key
      email = email_usuario
  ) Then
    INSERT INTO usuario (nombre, apellido, email, telefono,
      cedula, contraseña, direccion_nombre, direccion, ciudad ) VALUES
(nombre, apellido,email_usuario,telefono, cedula, contraseña, direccion_nombre, direccion, ciudad);

    id:=(select id_usuario from usuario where email=email_usuario);
    INSERT INTO trabajador(id_usuario, foto_perfil, estado_trabajador, cuenta_trabajador) VALUES
(id, foto_perfil, TRUE, cuenta_trabajador);

    id_trabajador_t:= (select id_trabajador from trabajador where id_usuario = id);
    INSERT INTO trabajador_labor(id_trabajador, id_labor, precio_labor, unidad_valor) VALUES
(id_trabajador_t, id_labor, precio, und);

  RETURN 1;
ELSE
  RAISE NOTICE 'No se puede ingresar';
RETURN 0;
End If;
End;
$$ Language plpgsql;
```

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

Para la validación del registro del cliente se implementó la siguiente función:

*Create or replace Function validar\_registro\_cliente( nombre VARCHAR(64), apellido VARCHAR(64), email\_usuario VARCHAR(64), telefono INTEGER, cedula INTEGER, contraseña VARCHAR(15), direccion\_nombre VARCHAR(64), direccion POINT, ciudad VARCHAR(64), recibo BYTEA, medio\_pago tipo\_pago)*

*Returns INTEGER*

*As \$\$*

*DECLARE*

*id integer;*

*Begin*

*If NOT Exists (*

*Select*

*\**

*From*

*usuario*

*Where*

*-- Assuming all three fields are primary key*

*email = email\_usuario*

*) Then*

*INSERT INTO usuario (nombre, apellido, email, telefono,*

*cedula, contraseña, direccion\_nombre, direccion, ciudad ) VALUES*

*(nombre, apellido,email\_usuario,telefono, cedula, contraseña, direccion\_nombre, direccion, ciudad);*

*id:=(select id\_usuario from usuario where email=email\_usuario);*

*INSERT INTO cliente(id\_usuario, recibo, medio\_pago) VALUES (id, NULL,medio\_pago);*

*RETURN 1;*

*ELSE*

*RAISE NOTICE 'No se puede ingresar';*

*RETURN 0;*

*End If;*

*End;*

*\$\$ Language plpgsql;*



Proyecto Bases de Datos 'Mande'  
Informe de Resultados y Cierre de Proyecto

Universidad del Valle

Una vez validados los registros, se pasa a validar el login de los trabajadores y de los clientes con la implementación de las siguientes funciones:

*Create or replace Function validar\_login\_trabajador(email\_usuario  
VARCHAR(64),contrasena\_usuario VARCHAR(15)) Returns INTEGER*

*As \$\$*

*Begin*

*If NOT Exists (*

*Select*

*\**

*From*

*usuario*

*Where*

*-- Assuming all three fields are primary key*

*email = email\_usuario*

*) Then*

*RAISE NOTICE 'No se puede ni validar';*

*RETURN 0;*

*End If;*

*IF exists(*

*select id\_trabajador*

*from trabajador*

*Natural join usuario*

*where email = email\_usuario and contraseña = contrasena\_usuario*

*) Then*

*RETURN 1;*


*ELSE*

*RETURN 0;*

*End If;*

*End;*

*\$\$ Language plpgsql;*

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

*Create or replace Function validar\_login\_cliente(email\_usuario VARCHAR(64),contrasena\_usuario VARCHAR(15)) Returns INTEGER*

*As \$\$*

*Begin*

*If NOT Exists (*

*Select*

*\**

*From*

*usuario*

*Where*

*-- Assuming all three fields are primary key*

*email = email\_usuario*

*) Then*

*RAISE NOTICE 'No se puede ni validar';*

*RETURN 0;*

*End If;*

*IF exists(*

*select id\_cliente*

*from cliente*

*Natural join usuario*

*where email = email\_usuario and contraseña = contrasena\_usuario*

*) Then*

*RETURN 1;*

*ELSE*

*RETURN 0;*

*End If;*

*End;*


*\$\$ Language plpgsql;*

---

Para cambiar el estado de un trabajador a 'ocupado' dependiendo del servicio que tome, se usa la siguiente función y su respectivo trigger:

---

*Create or replace Function cambio\_estado\_ocupado() Returns trigger*

	<p>Proyecto Bases de Datos ‘Mande’ Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	---	------------------------------

*As \$\$*

*DECLARE*

*id integer;*

*Begin*

```
id:=(select t.id_trabajador
      from trabajador as t
      NATURAL JOIN trabajador_labor as tl
      where tl.id_trabajador_labor = NEW.id_trabajador_labor );
```

*UPDATE trabajador SET estado\_trabajador= FALSE where id\_trabajador= id;*

*RETURN NULL;*

*End;*

*\$\$ Language plpgsql;*

*create trigger cambio\_estado BEFORE INSERT ON servicio FOR EACH ROW EXECUTE  
PROCEDURE cambio\_estado\_ocupado();*

---

Asimismo, se implementó la siguiente función y trigger para pasar a estado ‘disponible’, una vez finalizado el servicio tomado:

---

*Create or replace Function cambio\_estado\_disponible() Returns trigger*

*As \$\$*

*DECLARE*

*id INTEGER;*

*fecha DATE;*

*Begin*


```
id:= (select t.id_trabajador
      from pago_servicio AS ps
      NATURAL JOIN servicio AS s
      NATURAL JOIN trabajador_labor as tl
      NATURAL JOIN trabajador as t
      WHERE ps.id_servicio = NEW.id_servicio);
```

*fecha:=(SELECT NOW()) AS fecha\_inicial);*

*UPDATE trabajador SET estado\_trabajador= TRUE where id\_trabajador= id;*

*UPDATE servicio SET estado\_servicio =FALSE where id\_servicio = NEW.id\_servicio;*

*UPDATE servicio SET fecha\_final = fecha where id\_servicio = NEW.id\_servicio;*

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

```
RETURN NULL;
End;
$$ Language plpgsql;
```

```
create trigger cambio_estado_disponible AFTER INSERT ON pago_servicio FOR EACH ROW
EXECUTE PROCEDURE cambio_estado_disponible();
```

---

Se implementó una función que permite a los clientes editar información de su perfil:

---

```
Create or replace Function actualizar_cliente( nombre_c VARCHAR(64),
email_usuario VARCHAR(64),
apellido_c VARCHAR(64),
contraseña_c VARCHAR(15),
direccion_nombre_c VARCHAR(64),
direccion_c POINT,
ciudad_c VARCHAR(64),
pago_c tipo_pago
)
```


Returns INTEGER

```
As $$
DECLARE
id integer;

Begin

UPDATE usuario
SET nombre= nombre_c, apellido= apellido_c, contraseña=contraseña_c, direccion_nombre=
direccion_nombre_c, direccion= direccion_c, ciudad= ciudad_c WHERE email= email_usuario;

id:=(select id_usuario from usuario where email=email_usuario);
UPDATE cliente SET medio_pago= pago_c WHERE id_usuario= id;
```

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

RETURN 1;

End;

\$\$ Language plpgsql;

--Crear servicios

Create or replace Function crear\_servicio( email\_usuario\_cliente VARCHAR(64), trabajador\_labor INTEGER)

Returns INTEGER

As \$\$

DECLARE

id integer;

fecha DATE;

Begin

id:=(select c.id\_cliente from cliente AS c  
NATURAL JOIN usuario AS u  
where u.email=email\_usuario\_cliente);

fecha:=(SELECT NOW() AS fecha\_inicial);

INSERT INTO servicio (id\_cliente, id\_trabajador\_labor, fecha\_inicio, fecha\_final, estado\_servicio)  
VALUES (id, trabajador\_labor, fecha , NULL , true);

RETURN 1;

End;

\$\$ Language plpgsql;

---

Se implementó una función que permite a los trabajadores editar información de su perfil:

---

*Create or replace Function actualizar\_trabajador( nombre\_t VARCHAR(64), email\_usuario VARCHAR(64), apellido\_t VARCHAR(64), contraseña\_t VARCHAR(15), direccion\_nombre\_t VARCHAR(64), direccion\_t POINT, ciudad\_t VARCHAR(64), cuenta\_trabajador\_t INTEGER)*


*Returns INTEGER*

*As \$\$*

*DECLARE*

*id integer;*



	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

*Begin*

```

UPDATE usuario
SET nombre= nombre_t, apellido= apellido_t, contraseña=contraseña_t, direccion_nombre=
direccion_nombre_t, direccion= direccion_t, ciudad= ciudad_t WHERE email= email_usuario;

id:=(select id_usuario from usuario where email=email_usuario);
UPDATE trabajador SET cuenta_trabajador= cuenta_trabajador_t WHERE id_usuario= id;

RETURN 1;

End;
$$ Language plpgsql;
```

---

El trabajador, una vez registrado, puede agregar más labores a su perfil, a través de la siguiente función, la cual también valida que el labor no esté repetido para un mismo trabajador:

---

```

Create or replace Function agregar_labor_trabajador( email_usuario VARCHAR(64),
                                                    id_labor_t INTEGER,
                                                    precio INT,
                                                    und unidad )
```

```


Returns INTEGER
As $$
DECLARE
id integer;
id_trabajador_t INTEGER;
```

```

Begin
If NOT Exists (
Select
id_labor
From
trabajador_labor
NATURAL JOIN trabajador AS t
NATURAL JOIN usuario AS u
Where
id_labor=id_labor_t AND u.email=email_usuario
) Then
```

```

id:=(select id_usuario from usuario where email=email_usuario);
```

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

```

id_trabajador_t:= (select id_trabajador from trabajador where id_usuario = id);

INSERT INTO trabajador_labor(id_trabajador, id_labor, precio_labor, unidad_valor) VALUES
(id_trabajador_t, id_labor_t, precio, und);

RETURN 1;
ELSE
    RAISE NOTICE 'No se puede ingresar';
RETURN 0;
End If;
End;
$$ Language plpgsql;

```

---

El cliente se encarga de crear servicios con el uso de la siguiente función:

---

```

Create or replace Function crear_servicio( email_usuario_cliente VARCHAR(64), trabajador_labor
INTEGER)
Returns INTEGER
As $$
DECLARE
id integer;
fecha DATE;

Begin


id:=(select c.id_cliente from cliente AS c
    NATURAL JOIN usuario AS u
    where u.email=email_usuario_cliente);

fecha:=(SELECT NOW() AS fecha_inicial);

INSERT INTO servicio (id_cliente, id_trabajador_labor, fecha_inicio, fecha_final, estado_servicio)
VALUES (id, trabajador_labor, fecha , NULL , true);

RETURN 1;
End;
$$ Language plpgsql;

```

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

---

El cliente, al terminar un servicio se encarga de generar los pagos a sus trabajadores:

---

*Create or replace Function crear\_pago( id\_servicio\_t INTEGER, puntaje\_t INTEGER, descripcion VARCHAR(100))*

*Returns INTEGER*

*As \$\$*

*DECLARE*

*fecha DATE;*

*Begin*

*fecha:=(SELECT NOW()) AS fecha\_inicial);*

*INSERT INTO PAGO\_SERVICIO(id\_servicio, fecha\_pago, puntaje\_trabajador, reseña) VALUES (id\_servicio\_t, fecha, puntaje\_t, descripcion);*

*RETURN 1;*

*End;*

*\$\$ Language plpgsql;*

---

## 4. EVALUACIÓN DEL PROYECTO

### 4.1. Aspectos positivos


La creación de la plataforma fomentó el autoaprendizaje, así como potenciar la investigación por aparte de cada integrante y poder combinar los descubrimientos en un mismo trabajo colectivo.

Durante la elaboración de la plataforma, se realizaron múltiples casos de prueba para confirmar el correcto funcionamiento del proyecto. También, se llevaron a cabo dichos casos de prueba cada que se implementaba cada una de las funcionalidades anteriormente mencionadas para verificar que estaban siendo declaradas de una manera idónea con la estructura y sintaxis de los lenguajes utilizados.

Ejemplos de casos de prueba:

*SELECT validar\_registro\_trabajador('Natalia','Lopez','nn@gmail.com',1,1193086608,'1','calle 1', POINT(3.43722, -76.5225),'Cali',NULL,1,1,30000,'dia');*

*SELECT validar\_registro\_trabajador('Juan','Perez','jj@gmail.com',2,78930608,'2','calle 2', POINT(3.43722, -76.5225),'Cali',NULL,2,2,35000,'dia');*

	<p>Proyecto Bases de Datos 'Mande'</p> <p>Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	--	------------------------------

*SELECT validar\_registro\_trabajador('Andres','Gomez','aa@gmail.com',3,32830408,'3','calle 3',  
POINT(6.217, -75.567),'Medellin',NULL,3,3,40000,'dia');*

*SELECT validar\_registro\_trabajador('Oscar','Medina','oo@gmail.com',4,53050008,'4','calle 4',  
POINT(3.53944, -76.30361),'Palmira',NULL,4,4,15000,'hora');*

*SELECT validar\_registro\_trabajador('Gabriela','giraldo','gg@gmail.com',5,113050112,'5','calle 5',  
POINT(3.53944, -76.30361),'Palmira',NULL,5,4,14000,'hora');*

*SELECT validar\_registro\_trabajador('Ximena','Perez','xx@gmail.com',6,1121203108,'6','calle 6',  
POINT(3.53944, -76.30361),'Palmira',NULL,6,6,11000,'hora');*

*SELECT validar\_registro\_trabajador('Pedro','Camacho','pp@gmail.com',7,107203998,'7','calle 7',  
POINT(3.43722, -76.5225),'Cali',NULL,7,7,5000,'hora');*

*SELECT validar\_registro\_trabajador('Kevin','Moreno','kk@gmail.com',8,11109928,'8','calle 8',  
POINT(3.43722, -76.5225),'Cali',NULL,8,2,30000,'dia');*

*SELECT validar\_registro\_trabajador('Stella','Agudelo','ss@gmail.com',9,234050028,'9','calle 9',  
POINT(6.217, -75.567),'Medellin',NULL,9,3,50000,'dia');*

*SELECT validar\_registro\_trabajador('Bibiana','Arango','bb@gmail.com',10,6050001,'10','calle 10',  
POINT(6.217, -75.567),'Medellin',NULL,10,8,35000,'dia');*

*SELECT validar\_registro\_trabajador('Victoria','Fajardo','vv@gmail.com',11,1105791,'11','calle 11',  
POINT(3.43722, -76.5225),'Cali',NULL,11,8,35000,'dia');*

*SELECT validar\_registro\_cliente('Carolain','Jimenez','cc@gmail.com',12,56720409,'12','calle 1 a',  
POINT(3.43722, -76.5225),'Cali',NULL,'debito');*


*SELECT validar\_registro\_cliente('Hernando','Lopez','hh@gmail.com',13,66721119,'13','calle 1 b',  
POINT(3.43722, -76.5225),'Cali',NULL,'credito');*

*SELECT validar\_registro\_cliente('Luisa','Lara','ll@gmail.com',14,66540418,'14','calle 3 a',  
POINT(6.217, -75.567),'Medellin',NULL,'credito');*

*SELECT validar\_registro\_cliente('Maria','Castro','mm@gmail.com',15,715540018,'15','calle 4 a',  
POINT(3.53944, -76.30361),'Palmira',NULL,'debito');*

*SELECT agregar\_labor\_trabajador('nn@gmail.com', 2, '10000', 'hora');*

*SELECT agregar\_labor\_trabajador('xx@gmail.com', 8, '50000', 'dia');*

	<p>Proyecto Bases de Datos ‘Mande’ Informe de Resultados y Cierre de Proyecto</p>	<p>Universidad del Valle</p>
---	---	------------------------------

*SELECT agregar\_labor\_trabajador('ss@gmail.com', 11, '60000', 'hora');*

*SELECT crear\_servicio( 'cc@gmail.com', 1);*

*SELECT crear\_servicio( 'cc@gmail.com', 2);*

*SELECT crear\_servicio( 'cc@gmail.com', 7);*

*SELECT crear\_servicio( 'hh@gmail.com', 11);*

*SELECT crear\_servicio( 'll@gmail.com', 3);*

*SELECT crear\_servicio( 'll@gmail.com', 10);*

*SELECT crear\_servicio( 'mm@gmail.com', 4);*

*SELECT crear\_pago( 1, 4, 'Muy bien ');*

*SELECT crear\_pago( 5, 2, 'Lo peor');*

#### 4.2. Aspectos negativos

Durante el desarrollo del proyecto, no se pudieron implementar muchas funcionalidades que se tenían programadas para la plataforma por cuestiones limitantes de tiempo.

Asimismo, los recursos brindados para el desarrollo del proyecto no fueron los idóneos puesto que la implementación de contenedores es una funcionalidad difícil en máquinas operadas por el sistema operativo Windows. Los desarrolladores del proyecto no disponían de máquinas con Linux, obligándolos a recurrir a alternativas como máquinas virtuales con particiones de Linux, o migrar totalmente a dicho sistema operativo, causando varios errores de compatibilidad o recursos físicos para las máquinas disponibles por cada uno de los integrantes.