



Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Monterrey

TC2005B.573

**Ciclo 3: Entrega Final**

Luis Alberto Rodríguez Mendoza	A00828335
Rodolfo Sandoval Schipper	A01720253
Tadeo Andrés Willis Lozano	A00829341
Andrea Garza	A00832444

19/10/2022

## ***Ciclo 3: Entrega Final***

### **a) Nombre del proyecto:**

Project +

### **b) Introducción/Tema Resolver:**

Construir un simulador computacional que reproduzca el comportamiento parcial de un sistema empresarial, económico, social, político, educativo. Este simulador permitirá el análisis de escenarios actuales o futuros para el apoyo en la toma de decisiones hacia la mejora de alguno de los procesos, o componentes del sistema. El simulador tiene una interfaz gráfica equiparable a un videojuego 2D. El sistema de software está basado en modelo cliente-servidor, implementado en web con capacidad de conexión a una base de datos relacional. Con visualización de indicadores de control basada en técnicas de videojuegos (tablero 2D) para el apoyo a la toma de decisiones en la solución de un problema específico del socio formador. Se trabajó para el socio formador una página web en donde el usuario como el administrador podrán entrar a cursos ya creados en la plataforma, el administrador tendrá la opción de poder crear cursos nuevos a su gusto.

### **Lista de requerimientos funcionales (uso de diagramas UML: casos de uso y actividad).**

- Controladores que permiten la navegación de los Views dependiendo de las acciones de los usuarios. (Mantenedor, Login, Register, Home y Cursos).
- Modelos que retornan la información de la base de datos. (Modelo Cursos, Login, y Register).
- Archivo que permita la conexión de NET CORE con la base de datos postgres.
- Archivo User Datos donde se ejecutan procedimientos de acuerdo al resultado del query en la base de datos.
- Entidades, Tablas, Funciones y Stored Procedures dentro de la Base de datos que permiten guardar valores y ejecutar procedimientos.

**Entidades:**

- Registrarse:
- Login (con columna admin TRUE o FALSE para verificar si tiene acceso administrador con una función).:
- Cursos:

**Functions Usuario:**

- FN\_Validación: función que retorna TRUE o FALSE validando que el usuario es administrador. Para acceder VIEW privado donde existen procedures eliminar, guardar, etc.
- FN\_Obtener (Usuario ID)
- FN\_Consultar
- FN\_Login

**Stored Procedures Usuario:**

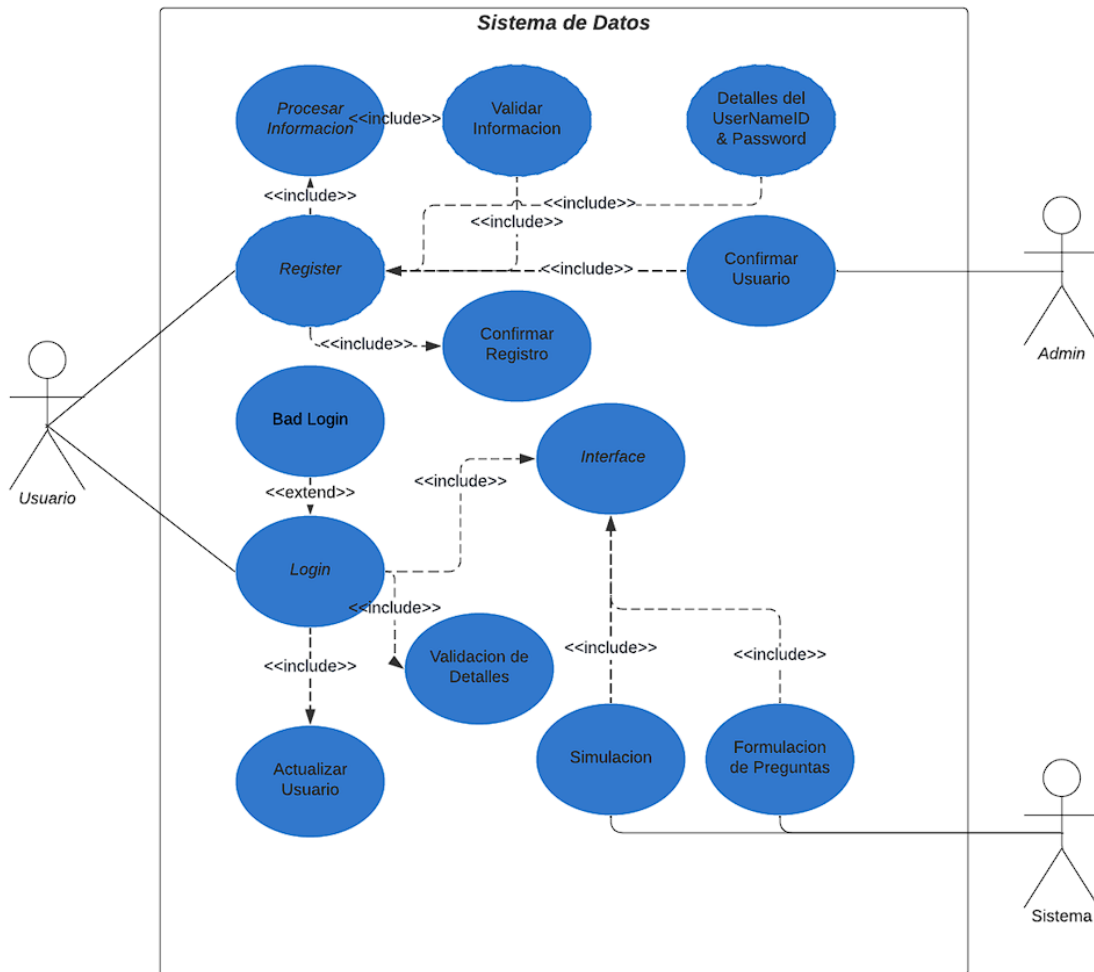
- SP\_GuardarUser
- SP\_EliminarUser
- SP\_ConsultarUser
- SP\_EditarUser

**Stored Procedures Cursos:**

- SP\_GuardarCourse

**Trigger Functions:**

- TF\_Bitácora: Observar Logs y movimientos del usuario (Si se crea o se elimina)



### Lista de requerimientos no funcionales:

- Botones interactivables en las vistas.
- Animaciones intuitivas para el usuario en la sección de Login.
- Colores similares a la paleta de IEPAM.
- Layout en la parte superior, que se actualiza en todas las plantillas.
- Required Fields en Registrarse y Login.
- Cartillas cuando se genera un Curso.
- Separación de Información en cada formato.
- Navegación con scroll.

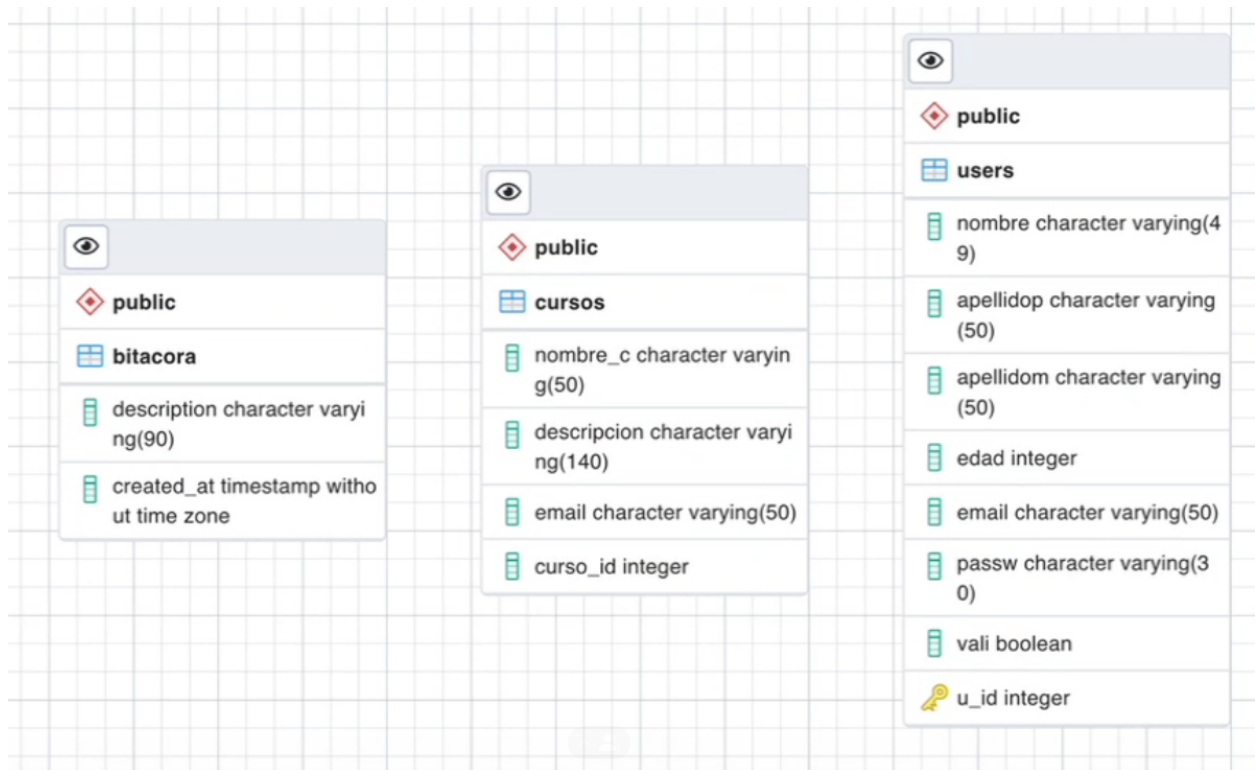
### **Definición de perfiles de usuario:**

- Registro de sus datos para la creación de la cuenta, entre ellos:
  - Nombre Completo
  - Edad
  - Correo electrónico
  - Contraseña
  
- Privilegios y limitaciones del usuario:
  - Acceso a los cursos dentro de la plataforma.
  - No poder modificar y/o crear cursos.
  
- Privilegios y limitaciones del usuario(administrador):
  - Acceso a los cursos dentro de la plataforma.
  - Acceso a modificar y/o crear cursos.
  - Acceso a modificar/agregar/eliminar datos del usuario.

### **Requisitos de despliegue del sistema:**

- Registro del Usuario y/o administrador.
- Verificación de la cuenta.
- Iniciar sesión para poder entrar a la sección de cursos.
- Seleccionar el curso deseado.
- Visualizar el contenido y aprender con las herramientas que brinda los cursos.

## Modelo relacional de la base de datos:



## Diagramas de estados:

Diagrama de Estado Login

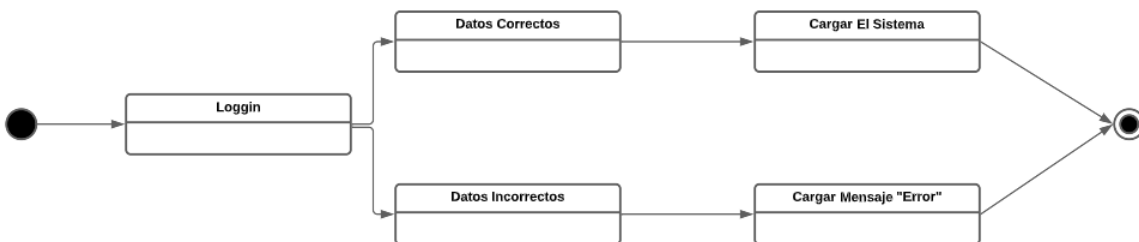
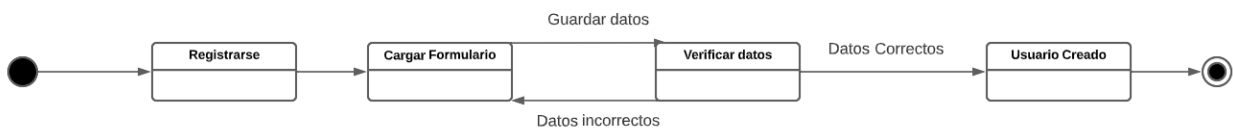
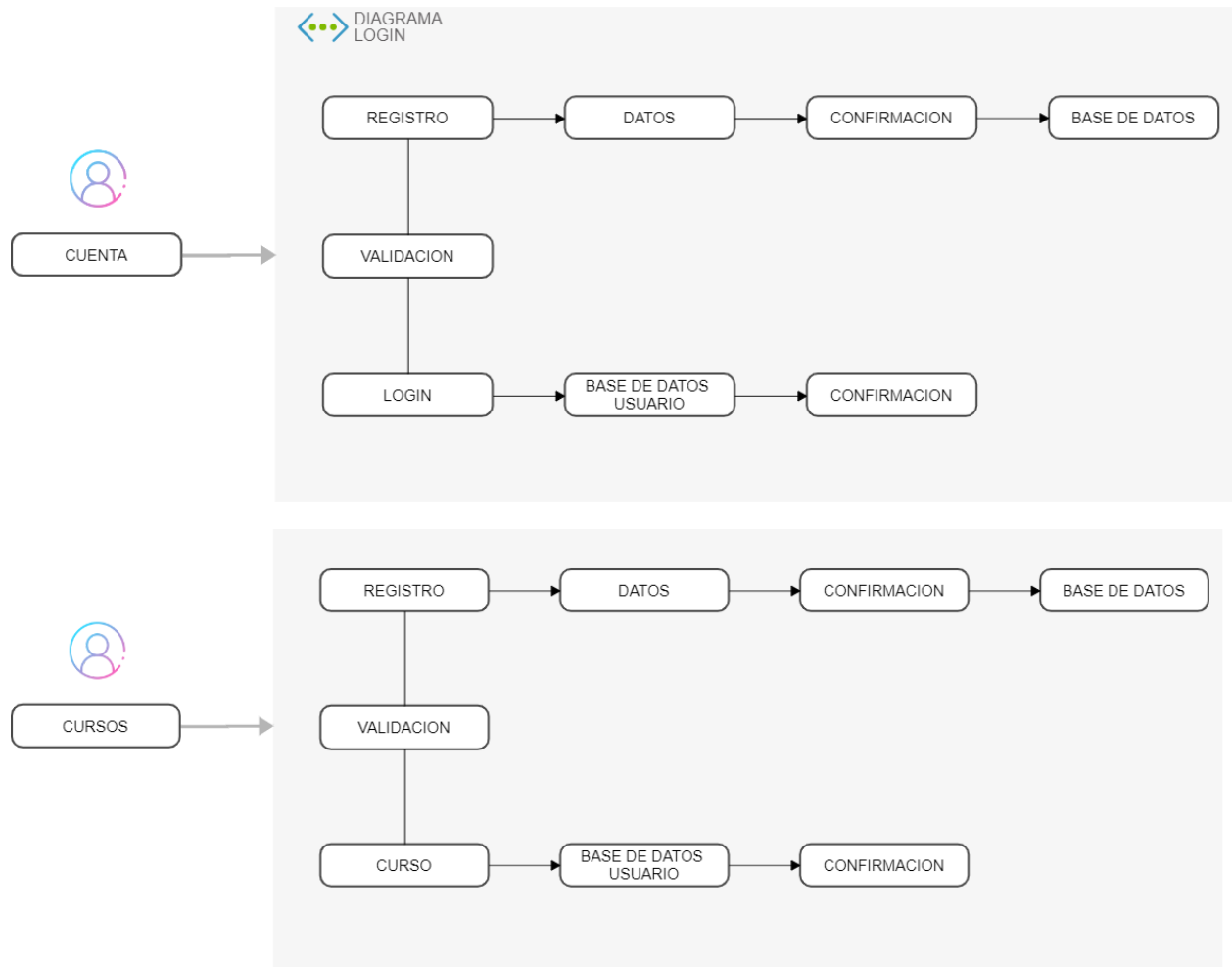


Diagrama de Estado Registro de usuario

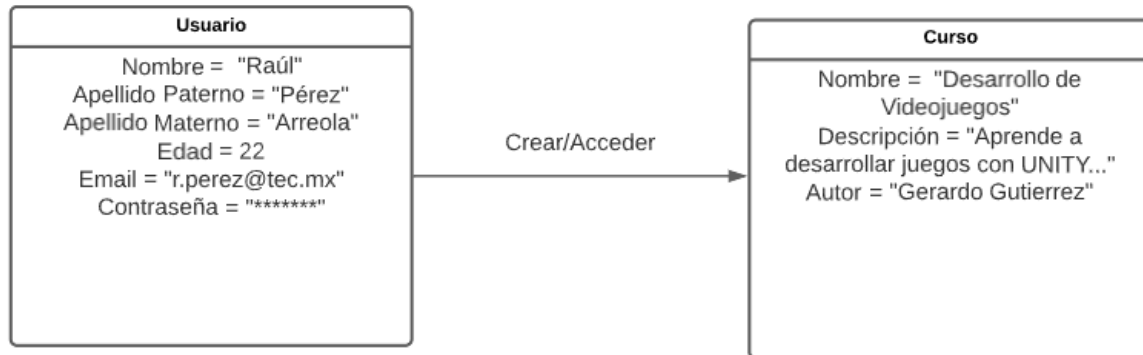


## Diagramas de componentes.

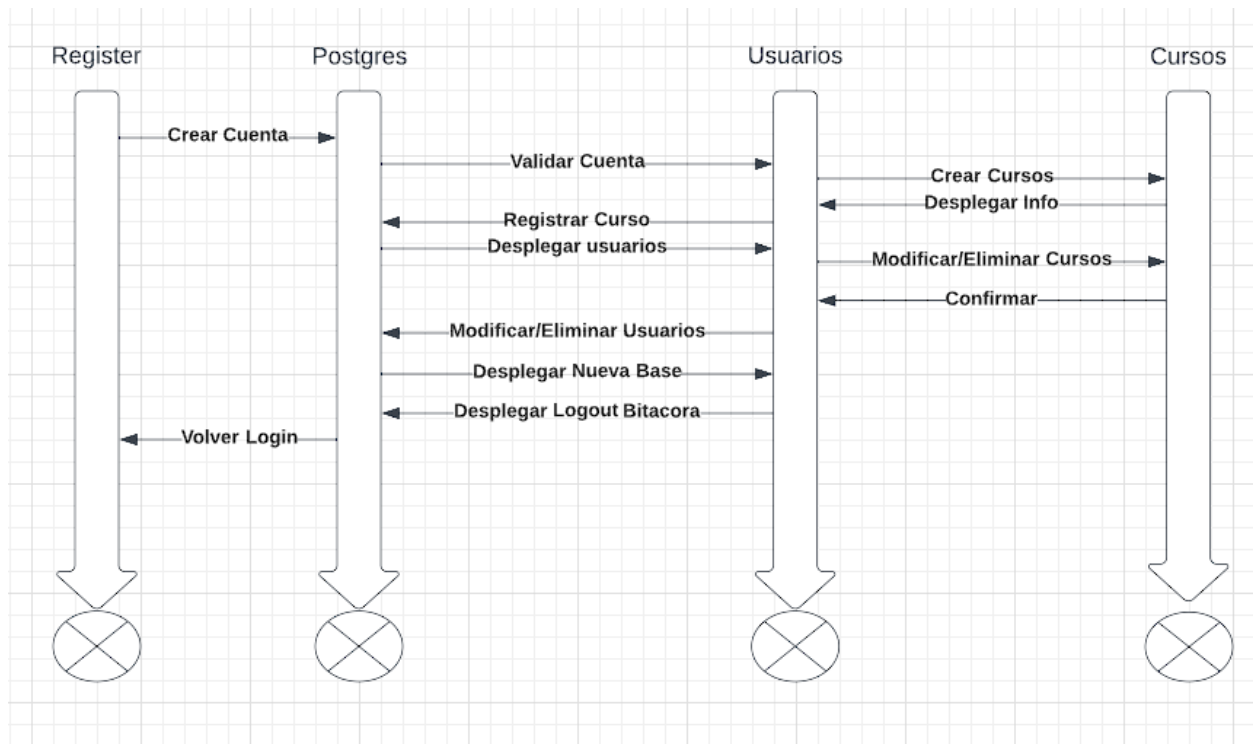


## Diagramas de objetos.

Diagrama de Objetos

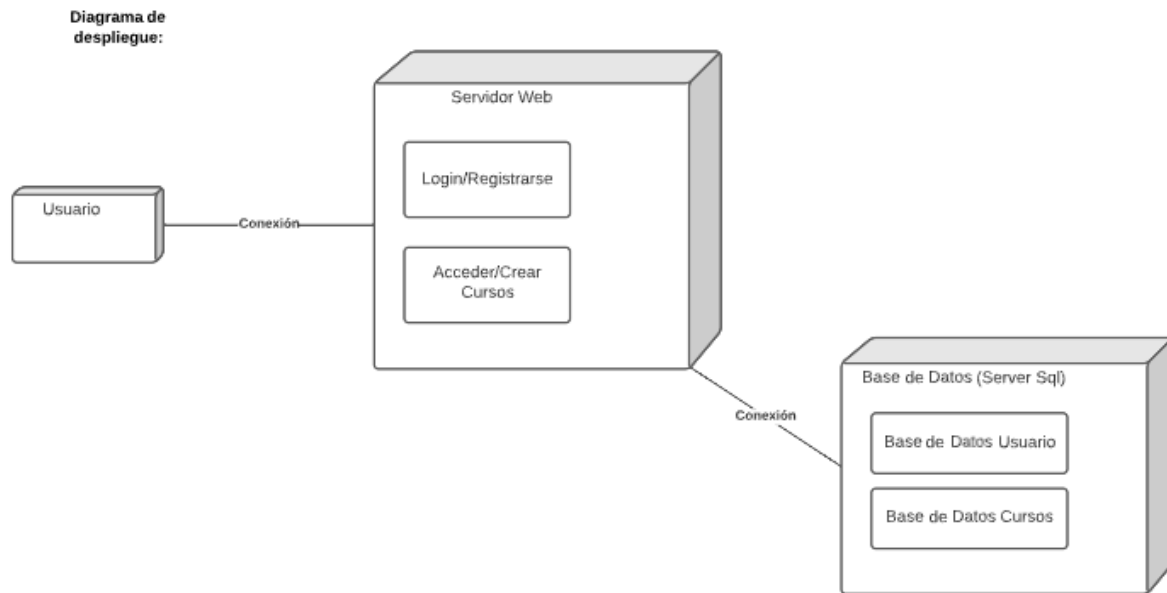


## Definición de pruebas estáticas y manuales:

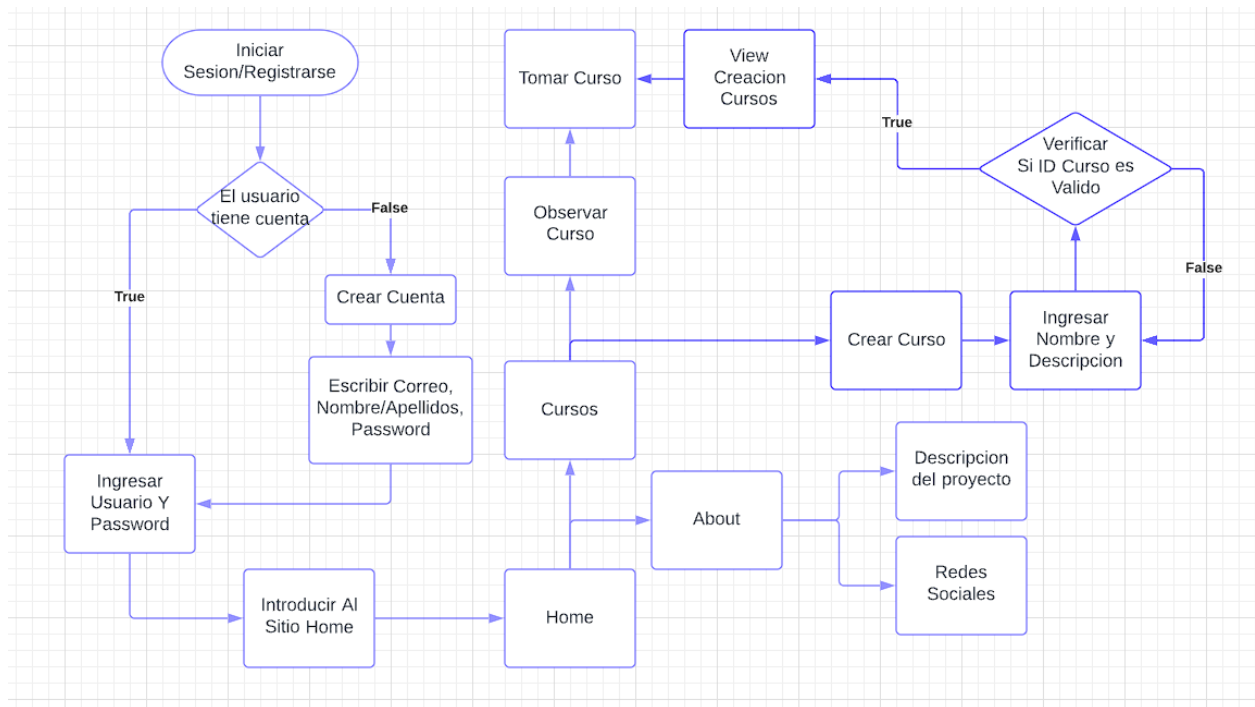




## Diagrama de despliegue:



## Diagrama de Interfaz :



## Definición de las pruebas unitarias:

The screenshot shows a contact form titled "Mandanos Un Mensaje" (Send us a message). On the left, there is contact information: "Dirección" (Address) as "Madrid 207, Mirador, 64070 Monterrey, N.L.", "Teléfono" (Phone) as "81 2723 0982" and "81 272 30 979", and "Correo" (Email) as "aulaiepam@nuevovoleon.gob.mx". The form itself has three input fields: "Introduce tu Nombre" (Enter your name), "Introduce tu Correo" (Enter your email), and "Introduce un Mensaje" (Enter a message). Below these fields is a purple "Enviar" (Send) button. The header of the page is dark purple with a logo on the left and navigation links: "Menu", "Cursos", "Conocenos", "Equipo", "Contacto", and "Log In".

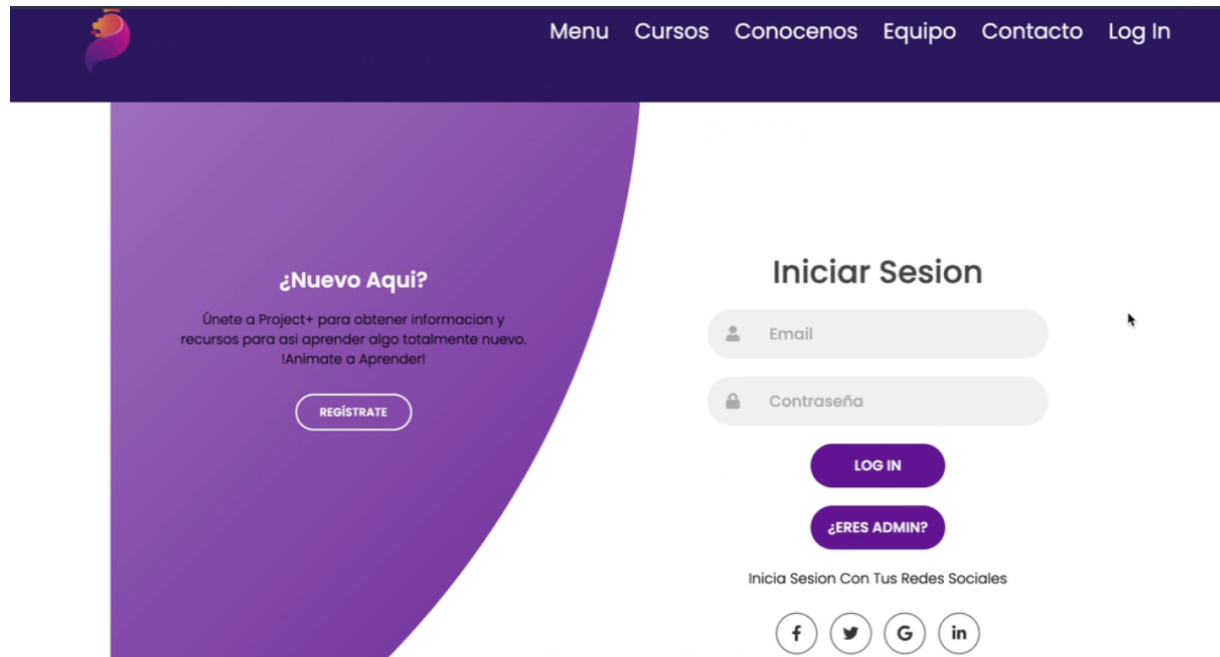
(CONTACT VIEW: En esta sección se visualizará una plantilla en donde podrás mandar tus dudas y/o comentarios que tengas para los socios formadores. )

## Definición de pruebas funcionales:

- VIEWS/Plantillas Funcionales:



(MENÚ VIEW: En esta sección aparece el título del proyecto como una descripción breve de lo que trata, de igual manera en la parte superior derecha se pueden observar las secciones de contenido de la página web.)



(LOGIN VIEW: El usuario ingresa con su cuenta que configuró al registrarse, esto permitirá el acceso a los cursos y se tomarán en cuenta las acciones del usuario)

**Regístrate**

Nombre

Apellido Paterno

Apellido Materno

edad

email

Contraseña

**GUARDAR**

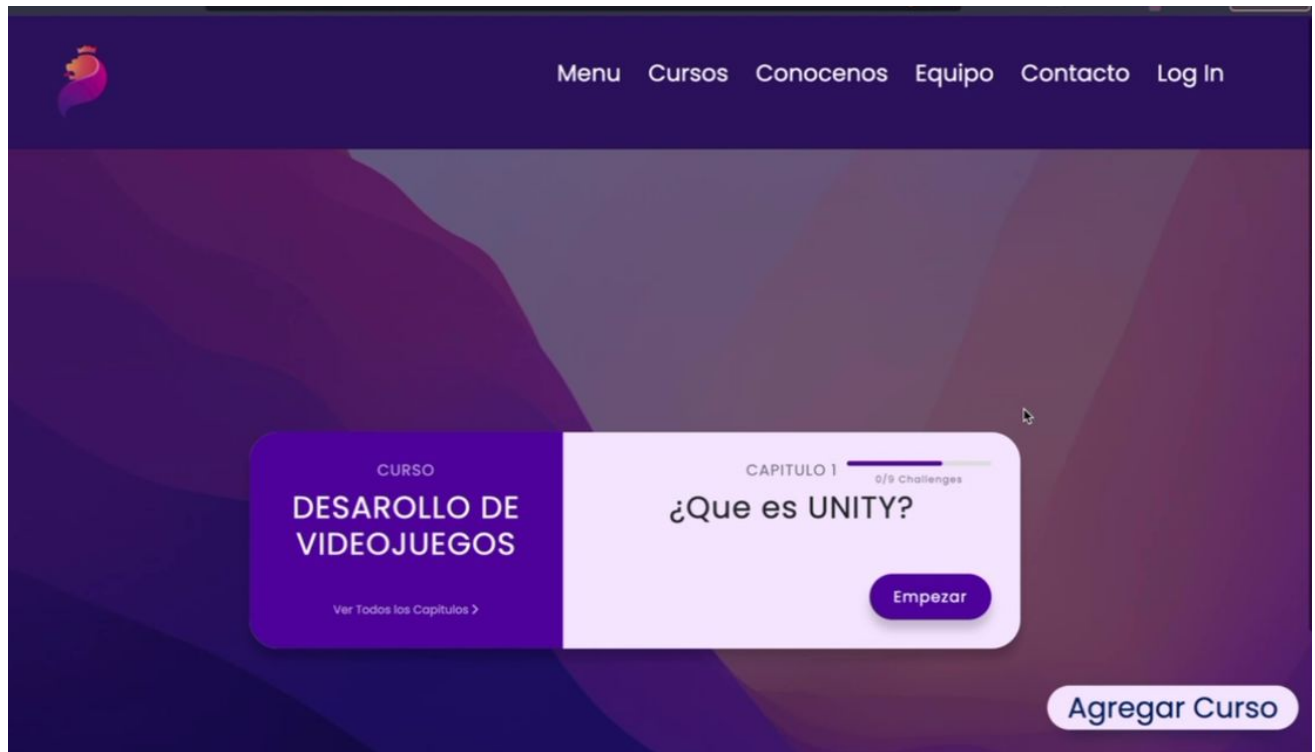
Regístrate Con Tus Redes Sociales

f t G in

**¿Ya Tienes Cuenta?**

INICIA SESIÓN

(REGISTER VIEW: En este apartado el usuario puede registrar su cuenta, donde la información será guardada en una base de datos organizada por tablas. Esto permitirá que esa cuenta pueda acceder por medio de la plantilla login para que finalmente el usuario pueda crear cursos. )

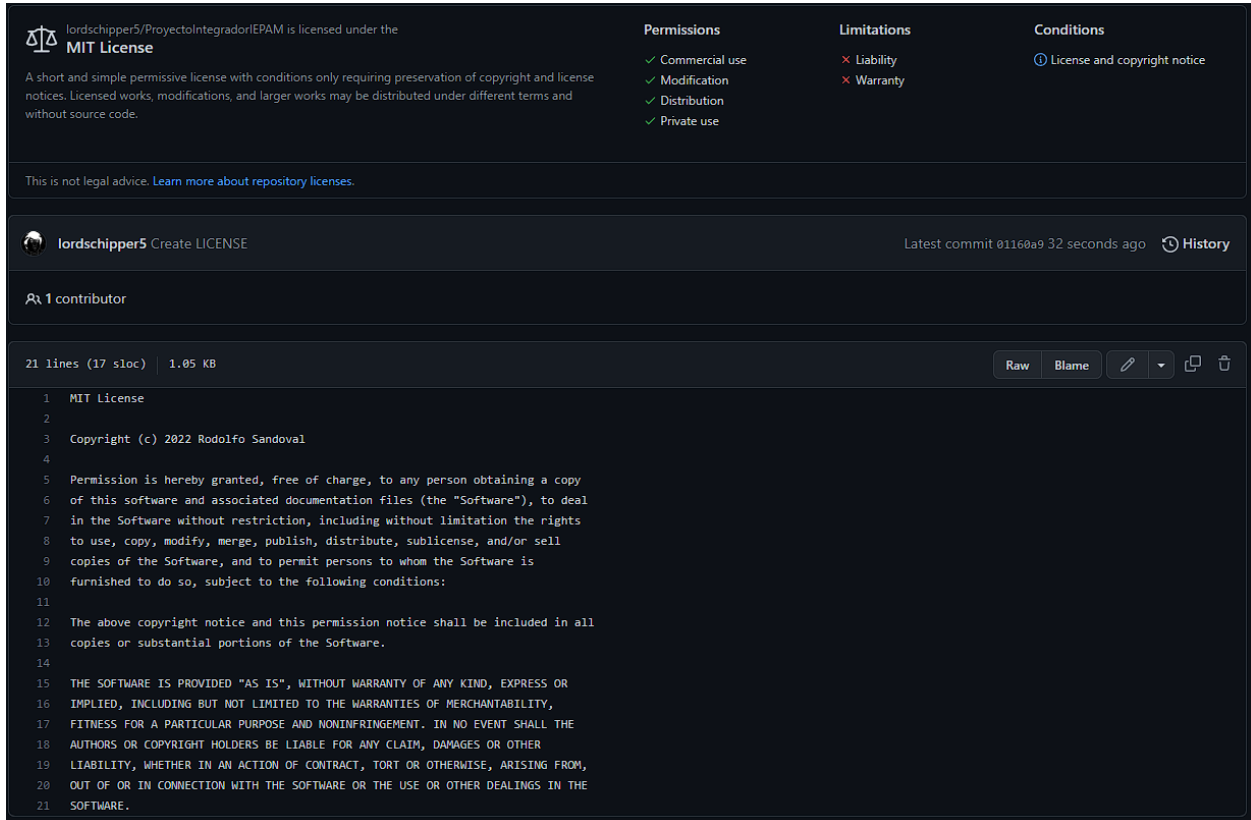


(COURSE VIEW: En esta sección se podrán visualizar los cursos que ya vienen incluidos en la plataforma, esté enseña el titulo como el tema a aprender y el avance que uno lleva dentro del curso. Posteriormente, en esta sección también se podrá crear cursos para los usuarios dentro de la plataforma.)



(CONTENTS VIEW: En esta sección se podrá visualizar el contenido del curso de videojuegos, esté enseña el título, los pasos a realizar para aprender y vídeos adicionales para un mejor aprendizaje. De igual manera, incluye el juego de ejemplo que podrías realizar con lo aprendido del curso..)

**Tipo de licencia de software que rige al producto, así como la identificación de derechos de autor y la clarificación (autoría y permisos) correspondiente en el uso de recursos externos (imágenes, diagramas, componentes, etc):**



The screenshot displays the MIT License for the repository 'lordschipper5/ProyectoIntegradorIEPAM'. It includes a summary of the license, a table of permissions and limitations, and the full text of the license.

Permissions	Limitations	Conditions
✓ Commercial use ✓ Modification ✓ Distribution ✓ Private use	✗ Liability ✗ Warranty	① License and copyright notice

```
1 MIT License
2
3 Copyright (c) 2022 Rodolfo Sandoval
4
5 Permission is hereby granted, free of charge, to any person obtaining a copy
6 of this software and associated documentation files (the "Software"), to deal
7 in the Software without restriction, including without limitation the rights
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in all
13 copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21 SOFTWARE.
```

*(Se utiliza la MIT License)*

## Elaborar el manual de usuario del proyecto (Proyecto)

Herramientas para ejecutar el software : Visual Studio (Net Core 3.1), Postgres (pgAdmin 4).

### Visual Studio.-

Se ejecuta la solución de WEBAPPCRUD-1 para visualizar el código fuente de la página.

### Postgres.-

Se ejecutan los Queries en una nueva Base de Datos (El archivo de Conexión.cs en visual studio debe fijarse a los mismos parámetros que tienes en postgres para ejecutar la página).

## **Manual de Usuario para la Navegación.-**

### **Menú.-**

Aquí el usuario puede navegar y observar información sobre la organización de IEPAM dentro del apartado [Home]. El usuario puede interactuar con la redes sociales y es presentado con las opciones LOGIN, CONTACTO, EQUIPO en el [Layout] en la parte superior.

### **Login/Registrar.-**

El usuario necesita registrarse primero antes de poder utilizar la función de Login. Es importante darle click en el botón de registrar en el apartado de login. La cual permitirá que el usuario ingrese su información para registrarla en la base de datos. Una vez que esté la cuenta registrada, nos enviará a la plantilla de cursos. Sin embargo, podemos regresar a la sección de login para poder acceder a los cursos nuevamente. Existen dos tipos de cuentas, Usuarios y Admin. Si tu cuenta es Usuario puedes ingresar directo a cursos. El administrador solo tiene acceso a la plantilla con la información de la base de datos.

Para acceder cómo administrador → **Generar un INSERT INTO en la base de datos TABLE users donde la columna vali = false. Esto hará que esa cuenta tenga acceso administrativo.**

### **Cursos.-**

El usuario podrá acceder a los cursos disponibles que se encuentran registrados en la base de datos. Al seleccionar un curso, el usuario podrá ver el contenido del curso en una nueva visualización. También, en este apartado, el usuario podrá introducir el título de un nuevo curso junto con su descripción al utilizar el botón de “Agregar curso” que se encuentra en la parte inferior derecha de la pantalla.

### **Contactos.-**

En este apartado, el usuario podrá enviar un mensaje directamente al correo del IEPAM rellenando la información en el formulario, en donde se le pedirá al usuario que ingrese su nombre, su correo y el contenido del mensaje que se quiere enviar.

### **Conocenos.-**

Esta sección te dirige a la página oficial de IEPAM, si el motivo es saber más de la organización. Este apartado nos permite visualizar más a profundidad.

### **Equipo.-**

En equipo se ubican los desarrolladores de la página, Base de datos y Juego. [Equipo 5 IEPAM].

**Crear un video en donde se muestre la funcionalidad del sistema, recorriendo los diferentes flujos de información para cada uno de los diferentes tipos de usuarios.**

Link del Video:

[https://drive.google.com/file/d/15Pfv3bS086sh0cuvxo7ym\\_w3W6qL9\\_E9/view?usp=sharing](https://drive.google.com/file/d/15Pfv3bS086sh0cuvxo7ym_w3W6qL9_E9/view?usp=sharing)

Github:

<https://github.com/lordschipper5/ProyectoIntegradorIEPAM.git>

**Queries (base De datos).-**

--TABLAS

```
CREATE TABLE users (  
    nombre varchar(49),  
    apellidop varchar(50),  
    apellidom varchar(50),  
    edad int,  
    email varchar(50),  
    passw varchar(30),  
    vali bool,  
    u_id serial PRIMARY KEY  
)
```

```
CREATE TABLE cursos (  
    nombre_c varchar(30),  
    descripcion varchar(255),  
    email varchar(50),  
    curso_id serial  
)
```

```
CREATE TABLE bitacora (  
    description VARCHAR(90) NOT NULL,  
    created_at TIMESTAMP DEFAULT NOW()
```



);

-- FIN DE LAS TABLAS

-- STORED PROCEDURES

```
CREATE OR REPLACE PROCEDURE user_edit (nombre varchar(49), apellidop
varchar(50), apellidom varchar(50), edad int, email varchar(50), passw varchar(30), u_id int)
    AS $$
    UPDATE users SET nombre = $1, apellidop = $2, apellidom = $3, edad = $4, email =
    $5, passw = $6
    WHERE u_id = $7
    $$ LANGUAGE SQL
```

```
CREATE OR REPLACE PROCEDURE user_del (u_id int)
    AS $$
    DELETE FROM users WHERE u_id = $1
    $$ LANGUAGE SQL
```

```
CREATE OR REPLACE PROCEDURE user_insert (nombre varchar(49), apellidop
varchar(50), apellidom varchar(50), edad int, email varchar(50), passw varchar(30), vali bool,
u_id int)
    AS $$
    INSERT INTO users VALUES(nombre, apellidop,apellidom,edad,email, passw,
vali,u_id)
    $$ LANGUAGE SQL
```

```
CREATE OR REPLACE PROCEDURE insert_course( nombre_c varchar(50), descripcion
varchar(140), email varchar(50))
    AS $$
    INSERT INTO cursos VALUES(nombre_c, descripcion, email)
```

**\$\$ LANGUAGE SQL**

**CREATE OR REPLACE PROCEDURE** guardar\_usuario (nombre **varchar**(49), apellidop **varchar**(50), apellidom **varchar**(50), edad **int**, email **varchar**(50), passw **varchar**(30), vali **bool**)  
**AS \$\$**  
**INSERT INTO** users **VALUES**(nombre, apellidop,apellidom,edad,email, passw, vali)  
**\$\$ LANGUAGE SQL**

-- FIN DE STORED PROCEDURES

-- INICIO DE FUNCTIONS

**CREATE OR REPLACE FUNCTION** fn\_validar(lg\_email **varchar**(50),lg\_passw **varchar**(30))  
**RETURNS TABLE**(bandera **int**)  
**AS \$\$**  
**SELECT COUNT** (email) **FROM** users **WHERE** email = lg\_email **AND** passw  
= lg\_passw  
**\$\$ LANGUAGE SQL**

**CREATE OR REPLACE FUNCTION** fn\_admin(lg\_email **varchar**(50), lg\_passw **varchar**(30),lg\_vali **bool**)  
**RETURNS TABLE**(admin\_vali **int**)  
**AS \$\$**  
**SELECT COUNT**(email) **FROM** users **WHERE** email = lg\_email **AND** passw  
= lg\_passw **AND** vali = lg\_vali  
**\$\$ LANGUAGE SQL**

**CREATE OR REPLACE FUNCTION** fn\_consultar ()  
**RETURNS TABLE**(nombre **varchar**(49), apellidop **varchar**(50), apellidom **varchar**(50),  
edad **int**, email **varchar**(50), passw **varchar**(30), vali **bool**, u\_id **int**)  
**AS \$\$**

```
        SELECT a.nombre, a.apellidop, a.apellidom, a.edad, a.email, a.passw, a.vali, a.u_id
FROM users AS a
    $$ LANGUAGE SQL
```

```
CREATE OR REPLACE FUNCTION fn_obtener(u_id int)
    RETURNS TABLE (nombre varchar(49), apellidop varchar(50), apellidom varchar(50),
edad int, email varchar(50), passw varchar(30), vali bool, u_id int)
    AS $$
        SELECT a.nombre, a.apellidop, a.apellidom, a.edad, a.email, a.passw, a.vali, a.u_id
FROM users AS a WHERE u_id = $1
    $$ LANGUAGE SQL
```

```
-- FIN DE FUNCIONES
```

```
-- INICIO TRIGGER FUNCTIONS
```

```
CREATE OR REPLACE FUNCTION fn_after_insert_user()
    RETURNS TRIGGER AS $BODY$
BEGIN
    INSERT INTO bitacora VALUES ('CREACION DE UN NUEVO USUARIO');
    RETURN NEW;
END $BODY$
LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION fn_after_delete_user()
    RETURNS TRIGGER AS $BODY$
BEGIN
    INSERT INTO bitacora VALUES ('ELIMINACIÓN DE UN USUARIO');
    RETURN NEW;
END $BODY$
LANGUAGE plpgsql;
```

```
CREATE TRIGGER tg_after_user  
AFTER INSERT ON users  
FOR EACH ROW  
EXECUTE PROCEDURE fn_after_insert_user();
```

```
CREATE TRIGGER tg_after_del_user  
AFTER DELETE ON users  
FOR EACH ROW  
EXECUTE PROCEDURE fn_after_delete_user();  
-- FIN TRIGGER FUNCTIONS
```

```
-- SELECTS DE PRUEBA  
SELECT * FROM users;  
SELECT * FROM bitacora;  
SELECT * FROM cursos;
```