Avance Final de Proyecto

Semana: 10

Integrantes del grupo:

Andrea Jimena Ortez Ramirez	12241140
Héctor Eduardo Hernández Pineda	12211096
Tatiana Zuseth Garcia Ferrufino	12241079
Fernando Jafet Castillo Palma	32211027

Asignatura:
Teoría de Bases de Datos I

Docente:
Ing. Julio Sandoval

Sección:
755

Fecha de entrega:
17 de marzo de 2024

Contenido

1. Introducción

En el vertiginoso mundo de los bienes raíces, la eficiencia y la organización son elementos esenciales para el éxito. En respuesta a esta necesidad, presentamos el proyecto final de Teoría de Bases de Datos I: una solución integral diseñada para optimizar las operaciones de una oficina de bienes raíces enfocada en el mercado residencial.

Este proyecto se centra en el desarrollo de una aplicación web que brinda soporte a las interacciones entre agentes, compradores, vendedores y las propiedades mismas. Conscientes de la importancia de una gestión eficaz, hemos diseñado una base de datos sólida y funcional, capaz de satisfacer las demandas específicas de esta industria dinámica.

La oficina de bienes raíces para la cual se ha desarrollado este sistema se destaca por su enfoque en viviendas, lo que implica una diversidad de transacciones y un flujo constante de información. Además, considerando el contexto de una administración con limitados conocimientos informáticos, nos hemos esforzado por ofrecer una interfaz intuitiva y fácil de usar, que permita una adopción rápida y sin complicaciones.

A través de este proyecto, no solo hemos creado una infraestructura robusta para la gestión de datos, sino que también hemos desarrollado una serie de consultas SQL que ilustran la eficacia y la versatilidad del sistema propuesto. Con la entrega de este proyecto, buscamos no solo cumplir con los requisitos establecidos, sino también superar las expectativas y brindar una solución integral que contribuya al éxito continuo de la oficina de bienes raíces.

2. Especificación

La aplicación consiste en las operaciones de una oficina de bienes raíces. La oficina necesita realizar un seguimiento de los agentes, compradores, vendedores, propiedades en el mercado y propiedades vendidas recientemente. Esta oficina se centra en viviendas más que en bienes raíces comerciales. La administración de esta agencia inmobiliaria no tiene muchos conocimientos de informática. Se le pide que diseñe la base de datos, la complete con datos de muestra (la administración no le permitirá realizar pruebas con datos en vivo debido a preocupaciones de privacidad) y escriba varias consultas SQL para demostrar el sistema. La oficina de bienes raíces está solicitando aproximadamente 50 propuestas, que evaluará a partir del 12 de abril de 2024.

3. Documentación y sistema

3.1. Diseño lógico del modelo relacional (DDL) en PostgreSQL

3.1.1. Creación de Tablas

```
CREATE TABLE AGENTS (
    IdentityNumber CHAR(13),
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(150) NOT NULL,
    PhoneNumber INTEGER NOT NULL,
    OfficePhone INTEGER NOT NULL,
    PRIMARY KEY (IdentityNumber),
    CHECK (PhoneNumber >= 10000000 AND PhoneNumber <= 99999999),
    CHECK (OfficePhone >= 10000000 AND OfficePhone <= 99999999)
);

CREATE TABLE SELLERS (
    IdentityNumber CHAR(13),
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(150) NOT NULL,
    PhoneNumber INTEGER NOT NULL,
    PRIMARY KEY (IdentityNumber),
    CHECK (PhoneNumber >= 10000000 AND PhoneNumber <= 99999999)
);

CREATE TABLE BUYERS (
    IdentityNumber CHAR(13),
    Name VARCHAR(50) NOT NULL,
    Address VARCHAR(150) NOT NULL,
    PhoneNumber INTEGER NOT NULL,
    PRIMARY KEY (IdentityNumber),
    CHECK (PhoneNumber >= 10000000 AND PhoneNumber <= 99999999)
```

```
);

CREATE TABLE PROPERTIESONMARKET (
    PropertyId SERIAL,
    Name VARCHAR(50) NOT NULL,
    City VARCHAR(20) NOT NULL,
    Address VARCHAR(150) NOT NULL,
    PhoneNumber INTEGER NOT NULL,
    BedroomCount INTEGER NOT NULL,
    Features VARCHAR(150) NOT NULL,
    Price REAL NOT NULL,
    PublicationDate DATE NOT NULL,
    AgentIdentityNumber CHAR(13) NOT NULL,
    SellerIdentityNumber CHAR(13) NOT NULL,
    image VARCHAR(255) NOT NULL,
    PRIMARY KEY (PropertyId),
    CONSTRAINT FK_AGENTS_MARKET FOREIGN KEY (AgentIdentityNumber)
REFERENCES AGENTS (IdentityNumber) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_SELLERS_MARKET FOREIGN KEY (SellerIdentityNumber)
REFERENCES SELLERS (IdentityNumber) ON DELETE CASCADE ON UPDATE CASCADE,
    CHECK (PhoneNumber >= 10000000 AND PhoneNumber <= 99999999)
);

CREATE TABLE SOLDPROPERTIES (
    PropertyId SERIAL,
    Name VARCHAR(50) NOT NULL,
    City VARCHAR(20) NOT NULL,
    Address VARCHAR(150) NOT NULL,
    PhoneNumber INTEGER NOT NULL,
    BedroomCount INTEGER NOT NULL,
    Features VARCHAR(150) NOT NULL,
    Price REAL NOT NULL,
    SalePrice REAL NOT NULL,
```

```
    PublicationDate DATE NOT NULL,
    SaleDate DATE NOT NULL,
    AgentIdentityNumber CHAR(13) NOT NULL,
    SellerIdentityNumber CHAR(13) NOT NULL,
    BuyerIdentityNumber CHAR(13) NOT NULL,
    SaleCommission REAL NOT NULL,
    image VARCHAR(255) NOT NULL,
    PRIMARY KEY (PropertyId),
    CONSTRAINT FK_AGENTS_SOLD FOREIGN KEY (AgentIdentityNumber)
        REFERENCES AGENTS (IdentityNumber) ON DELETE CASCADE ON UPDATE
CASCADE,
    CONSTRAINT FK_SELLERS_SOLD FOREIGN KEY (SellerIdentityNumber)
        REFERENCES SELLERS (IdentityNumber) ON DELETE CASCADE ON UPDATE
CASCADE,
    CONSTRAINT FK_BUYERS_SOLD FOREIGN KEY (BuyerIdentityNumber)
        REFERENCES BUYERS (IdentityNumber) ON DELETE CASCADE ON UPDATE
CASCADE,
    CHECK (PhoneNumber >= 10000000 AND PhoneNumber <= 99999999),
    CHECK (PublicationDate < SaleDate)
);

CREATE TABLE USERS (
    IdentityNumber CHAR(13) PRIMARY KEY,
    Username VARCHAR(20) UNIQUE,
    Password VARCHAR(20) NOT NULL,
    isAdmin BOOLEAN NOT NULL,
    isSeller BOOLEAN NOT NULL,
    isBuyer BOOLEAN NOT NULL,
    isAgent BOOLEAN NOT NULL
);

CREATE TABLE BITACORA (
id_bitacora serial primary key,
```

```
    action VARCHAR(150) NOT NULL,
        username varchar(20) NOT NULL,
        date date NOT NULL,
        time time NOT NULL
);
```

3.1.2. Procedimiento Almacenados (CRUDs)

```
CREATE OR REPLACE PROCEDURE public.pdeleteagent(
        IN in_identitynumber character)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    DELETE FROM agents
        WHERE identitynumber = in_IdentityNumber;
END;
$BODY$;
ALTER PROCEDURE public.pdeleteagent(character)
    OWNER TO fer;


CREATE OR REPLACE PROCEDURE public.pdeletepom(
        IN in_propertyid integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    DELETE FROM PropertiesOnMarket
        WHERE propertyid = in_PropertyID;
END;
$BODY$;
ALTER PROCEDURE public.pdeletepom(integer)
    OWNER TO fer;
```

```
CREATE OR REPLACE PROCEDURE public.pdeleteseller(
	IN in_identitynumber character)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
   DELETE FROM sellers
	WHERE identitynumber = in_IdentityNumber;
END;
$BODY$;
ALTER PROCEDURE public.pdeleteseller(character)
   OWNER TO fer;


CREATE OR REPLACE PROCEDURE public.pdeletesp(
	IN in_propertyid integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
   DELETE FROM SoldProperties
	WHERE propertyid = in_PropertyID;
END;
$BODY$;
ALTER PROCEDURE public.pdeletesp(integer)
   OWNER TO fer;


CREATE OR REPLACE PROCEDURE public.pinsertagent(
	IN identitynumber character,
	IN name character varying,
	IN address character varying,
```

```
        IN phonenumber integer,
        IN officephone integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
   INSERT INTO agents (IdentityNumber, Name, Address, PhoneNumber, OfficePhone)
   VALUES (IdentityNumber, Name, Address, PhoneNumber, OfficePhone);
END;
$BODY$;
ALTER PROCEDURE public.pinsertagent(character, character varying, character varying, integer, integer)
   OWNER TO fer;


CREATE OR REPLACE PROCEDURE public.pinsertpom(
        IN name character varying,
        IN city character varying,
        IN address character varying,
        IN phonenumber integer,
        IN bedroomcount integer,
        IN features character varying,
        IN price real,
        IN publicationdate date,
        IN agentidentitynumber character,
        IN selleridentitynumber character,
        IN image character varying)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
   INSERT INTO PropertiesOnMarket (name, city, address, phonenumber, bedroomcount, features, price,publicationdate, agentidentitynumber, selleridentitynumber, image)
   VALUES (name, city, address, phonenumber, bedroomcount, features, price,publicationdate, agentidentitynumber, selleridentitynumber, image);
```

```
END;
$BODY$;
ALTER PROCEDURE public.pinsertpom(character varying, character varying, character varying, integer, integer, character varying, real, date, character, character, character varying)
    OWNER TO fer;



CREATE OR REPLACE PROCEDURE public.pinsertseller(
        IN identitynumber character,
        IN name character varying,
        IN address character varying,
        IN phonenumber integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    INSERT INTO sellers (IdentityNumber, Name, Address, PhoneNumber)
    VALUES (IdentityNumber, Name, Address, PhoneNumber);
END;
$BODY$;
ALTER PROCEDURE public.pinsertseller(character, character varying, character varying, integer)
    OWNER TO fer;



CREATE OR REPLACE PROCEDURE public.pinsertsp(
        IN p_propertyId integer,
        IN name character varying,
        IN city character varying,
        IN address character varying,
        IN phonenumber integer,
        IN bedroomcount integer,
        IN features character varying,
        IN price real,
        IN saleprice real,
```

```
        IN publicationdate date,
        IN saledate date,
        IN agentidentitynumber character,
        IN selleridentitynumber character,
        IN buyeridentitynumber character,
        IN salecommission real,
        IN image character varying)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    INSERT INTO SoldProperties (name, city, address, phonenumber, bedroomcount, features, price, saleprice, publicationdate, saledate, agentidentitynumber, selleridentitynumber, buyeridentitynumber, salecommission, image)
    VALUES (name, city, address, phonenumber, bedroomcount, features, price, saleprice, publicationdate, saledate, agentidentitynumber, selleridentitynumber, buyeridentitynumber, salecommission, image);
        CALL pdeletepom(p_propertyId);
END;
$BODY$;
ALTER PROCEDURE public.pinsertsp(integer, character varying, character varying, character varying, integer, integer, character varying, real, real, date, date, character, character, character, real, character varying)
    OWNER TO fer;


CREATE OR REPLACE PROCEDURE public.pmodifyagent(
        IN in_identitynumber character,
        IN in_name character varying,
        IN in_address character varying,
        IN in_phonenumber integer,
        IN in_officephone integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    UPDATE agents
```

```
    SET
        name = in_Name,
        address = in_Address,
                    phoneNumber = in_PhoneNumber,
        officephone = in_OfficePhone
    WHERE identitynumber = in_IdentityNumber;
END;
$BODY$;
ALTER PROCEDURE public.pmodifyagent(character, character varying, character varying, integer,
integer)
    OWNER TO fer;



CREATE OR REPLACE PROCEDURE public.pmodifypom(
            IN in_propertyid integer,
            IN in_name character varying,
            IN in_city character varying,
            IN in_address character varying,
            IN in_phonenumber integer,
            IN in_bedroomcount integer,
            IN in_features character varying,
            IN in_price real,
            IN in_publicationdate date,
            IN in_agentidentitynumber character,
            IN in_selleridentitynumber character,
            IN in_image character varying)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    UPDATE PropertiesOnMarket
    SET
        name = in_name,
                    city = in_city,
```

```
    address = in_address,
                phoneNumber = in_phonenumber,
                bedroomcount = in_bedroomcount,
                features = in_features,
                price = in_price,
                publicationdate = in_publicationdate,
                agentidentitynumber = in_agentidentitynumber,
                selleridentitynumber = in_selleridentitynumber,
                image = in_image
  WHERE propertyid = in_propertyid;
END;
$BODY$;
ALTER PROCEDURE public.pmodifypom(integer, character varying, character varying, character varying, integer, integer, character varying, real,date, character, character, character varying)
  OWNER TO fer;



CREATE OR REPLACE PROCEDURE public.pmodifyseller(
        IN in_identitynumber character,
        IN in_name character varying,
        IN in_address character varying,
        IN in_phonenumber integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
  UPDATE sellers
  SET
    name = in_name,
    address = in_address,
                phoneNumber = in_phonenumber
  WHERE identitynumber = in_identitynumber;
END;
$BODY$;
```

```
ALTER PROCEDURE public.pmodifyseller(character, character varying, character varying, integer)
    OWNER TO fer;


CREATE OR REPLACE PROCEDURE public.pmodifysp(
            IN in_propertyid integer,
            IN in_name character varying,
            IN in_city character varying,
            IN in_address character varying,
            IN in_phonenumber integer,
            IN in_bedroomcount integer,
            IN in_features character varying,
            IN in_price real,
            IN in_saleprice real,
            IN in_publicationdate date,
            IN in_saledate date,
            IN in_agentidentitynumber character,
            IN in_selleridentitynumber character,
            IN in_buyeridentitynumber character,
            IN in_salecommission real,
            IN in_image character varying)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    UPDATE SoldProperties
    SET
        name = in_name,
                    city = in_city,
        address = in_address,
                    phoneNumber = in_phonenumber,
                    bedroomcount = in_bedroomcount,
                    features = in_features,
                    price = in_price,
```

```
            saleprice = in_saleprice,
            publicationdate = in_publicationdate,
            saledate = in_saledate,
            agentidentitynumber = in_agentidentitynumber,
            selleridentitynumber = in_selleridentitynumber,
            buyeridentitynumber = in_buyeridentitynumber,
            salecommission = in_salecommission,
            image = in_image
    WHERE propertyid = in_propertyid;
END;
$BODY$;
ALTER PROCEDURE public.pmodifysp(integer, character varying, character varying, character
varying, integer, integer, character varying, real, real,date, date, character, character, character, real,
character varying)
    OWNER TO fer;

CREATE OR REPLACE PROCEDURE public.pinsertbuyer(
        IN identitynumber character,
        IN name character varying,
        IN address character varying,
        IN phonenumber integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    INSERT INTO buyers (IdentityNumber, Name, Address, PhoneNumber)
    VALUES (IdentityNumber, Name, Address, PhoneNumber);
END;
$BODY$;
ALTER PROCEDURE public.pinsertbuyer(character, character varying, character varying, integer)
    OWNER TO fer;

CREATE OR REPLACE PROCEDURE public.pdeletebuyer(
        IN in_identitynumber character)
```

```
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
   DELETE FROM buyers
         WHERE identitynumber = in_IdentityNumber;
END;
$BODY$;
ALTER PROCEDURE public.pdeletebuyer(character)
   OWNER TO fer;

CREATE OR REPLACE PROCEDURE public.pmodifybuyer(
         IN in_identitynumber character,
         IN in_name character varying,
         IN in_address character varying,
         IN in_phonenumber integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
   UPDATE buyers
   SET
      name = in_name,
      address = in_address,
                  phoneNumber = in_phonenumber
   WHERE identitynumber = in_identitynumber;
END;
$BODY$;
ALTER PROCEDURE public.pmodifybuyer(character, character varying, character varying, integer)
   OWNER TO fer;
```

3.1.3. Vistas/Consultas

1. **Cantidad de ventas por agente.**

```
CREATE OR REPLACE VIEW public.vsalesagent
 AS
 SELECT ag.identitynumber,
    ag.name,
    count(sp.agentidentitynumber) AS amount_of_sales_per_agent
   FROM agents ag
     LEFT JOIN soldproperties sp ON ag.identitynumber = sp.agentidentitynumber
  GROUP BY ag.identitynumber;

ALTER TABLE public.vsalesagent
    OWNER TO fer;
```

2. Cantidad de ventas por vendedor.

```
CREATE OR REPLACE VIEW public.vsalesseller
 AS
 SELECT se.identitynumber,
    se.name,
    count(sp.selleridentitynumber) AS amount_of_sales_per_seller
   FROM sellers se
     LEFT JOIN soldproperties sp ON se.identitynumber = sp.selleridentitynumber
  GROUP BY se.identitynumber;

ALTER TABLE public.vsalesseller
    OWNER TO fer;
```

3. Cantidad de compras por comprador.

```
CREATE OR REPLACE VIEW public.vbuyerpurchases
 AS
 SELECT bu.identitynumber,
    bu.name,
    count(sp.buyeridentitynumber) AS amount_of_sales_per_buyer
   FROM buyers bu
```

```
    LEFT JOIN soldproperties sp ON bu.identitynumber = sp.buyeridentitynumber
  GROUP BY bu.identitynumber;

ALTER TABLE public.vbuyerpurchases
    OWNER TO fer;
```

4. **Ventas por ubicación.**

```
CREATE OR REPLACE VIEW public.vcitysales
 AS
 SELECT city,
    count(*) AS count
   FROM soldproperties sp
  GROUP BY city;

ALTER TABLE public.vcitysales
    OWNER TO fer;
```

5. **Ventas por precio de propiedad.**

```
CREATE OR REPLACE VIEW public.vsalesprice
 AS
 SELECT saleprice,
    count(*) AS count
   FROM soldproperties sp
  GROUP BY saleprice;

ALTER TABLE public.vsalesprice
    OWNER TO fer;
```

6. **Ventas de propiedades por características (por ejemplo, cantidad de habitaciones, si tiene piscina, etc.).**

```
CREATE OR REPLACE VIEW public.vsalesfeature
 AS
 SELECT bedroomcount,
```

```
    propertyid,
    name,
        CASE
            WHEN lower(features::text) ~~ '%have pool%'::text THEN 'Yes'::text
            ELSE 'No'::text
        END AS have_pool
   FROM soldproperties
  ORDER BY bedroomcount;


ALTER TABLE public.vsalesfeature
    OWNER TO fer;
```

7. Agente que vendió la mayor cantidad de propiedades en el año por valor total.

```
CREATE OR REPLACE FUNCTION public.fbestsellingagent(
            useryear integer)
    RETURNS TABLE(name character varying, identitynumber character, sale_year numeric,
total_sale real)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
    RETURN QUERY
    SELECT vBestSellingAgent.name, vBestSellingAgent.identitynumber,
vBestSellingAgent.saleyear, vBestSellingAgent.total_sale
    FROM vBestSellingAgent
    WHERE useryear = saleyear
    LIMIT 1;
END;
$BODY$;
```

```
ALTER FUNCTION public.fbestsellingagent(integer)
    OWNER TO fer;



CREATE OR REPLACE VIEW public.vbestsellingagent
 AS
 SELECT agents.name,
    agents.identitynumber,
    EXTRACT(year FROM sp.saledate) AS saleyear,
    sum(sp.saleprice) AS total_sale
   FROM agents
     JOIN SoldProperties sp ON agents.identitynumber = sp.agentidentitynumber
  GROUP BY agents.identitynumber, (EXTRACT(year FROM sp.saledate))
  ORDER BY (EXTRACT(year FROM sp.saledate)), (sum(sp.saleprice)) DESC;

ALTER TABLE public.vbestsellingagent
    OWNER TO fer;
```

8. **Para cada agente, calcule el precio de venta promedio de las propiedades vendidas en el año y el tiempo promedio que la propiedad estuvo en el mercado.**

```
CREATE OR REPLACE FUNCTION public.fagentsperformance(
    IN useryear integer
)
RETURNS TABLE (
            name character varying (50),
    identitynumber character(13),
    sale_year numeric,
            average_selling_price double precision,
    average_time_on_market numeric
) AS $BODY$
BEGIN
```

```
    RETURN QUERY
    SELECT vAgentsPerformance.name, vAgentsPerformance.identitynumber,
vAgentsPerformance.saleyear, vAgentsPerformance.average_selling_price,
vAgentsPerformance.average_time_on_market
    FROM vAgentsPerformance
    WHERE useryear = saleyear;
END;
$BODY$
LANGUAGE 'plpgsql';

ALTER FUNCTION public.fagentsperformance(integer)
    OWNER TO fer;

CREATE OR REPLACE VIEW public.vagentsperformance
 AS
 SELECT agents.name,
    agents.identitynumber,
    EXTRACT(year FROM sp.saledate) AS saleyear,
    avg(sp.saleprice) AS average_selling_price,
    avg(sp.saledate - sp.publicationdate) AS average_time_on_market
   FROM agents
     JOIN SoldProperties sp ON agents.identitynumber = sp.agentidentitynumber
  GROUP BY agents.identitynumber, (EXTRACT(year FROM sp.saledate))
  ORDER BY (EXTRACT(year FROM sp.saledate)), (sum(sp.saleprice));

ALTER TABLE public.vagentsperformance
    OWNER TO fer;
```

3.1.4. Otras vistas

1. **Todas las propiedades en el mercado que tiene un agente**

```
CREATE OR REPLACE VIEW public.vallpomAgent
 AS
```

```
 SELECT pom.propertyid,
    pom.name,
    pom.address,
    pom.city,
    pom.phonenumber,
    pom.bedroomcount,
    pom.features,
    pom.price,
    to_char(pom.publicationdate::timestamp with time zone, 'YYYY-MM-DD'::text) AS
publicationdate,
    image,
            ag.name as agentName,
            ag.OfficePhone as agentNumber
   FROM PropertiesOnMarket pom
   INNER JOIN agents ag
   ON ag.IdentityNumber= pom.AgentIdentityNumber;


ALTER TABLE public.vallpomAgent
    OWNER TO fer;
```

2. Enseña todas las propiedades en el mercado

```
CREATE OR REPLACE VIEW public.vallpom
 AS
 SELECT propertyid,
    name,
    address,
    city,
    phonenumber,
    bedroomcount,
    features,
    price,
    to_char(publicationdate::timestamp with time zone, 'YYYY-MM-DD'::text) AS publicationdate,
```

```
    agentidentitynumber,
    selleridentitynumber,
    image
   FROM PropertiesOnMarket;

ALTER TABLE public.vallpom
    OWNER TO fer;
```

3. Enseña todas las propiedades vendidas

```
CREATE OR REPLACE VIEW public.vallsp
 AS
 SELECT propertyid,
    name,
    address,
    city,
    phonenumber,
    bedroomcount,
    features,
    price,
    saleprice,
    to_char(publicationdate::timestamp with time zone, 'YYYY-MM-DD'::text) AS publicationdate,
    to_char(saledate::timestamp with time zone, 'YYYY-MM-DD'::text) AS saledate,
    agentidentitynumber,
    selleridentitynumber,
    buyeridentitynumber,
    salecommission,
    image
   FROM soldproperties;

ALTER TABLE public.vallsp
    OWNER TO fer;
```

4. Enseña todos los compradores

```
CREATE OR REPLACE VIEW public.vallbuyers
 AS
 SELECT identitynumber,
    name,
    address,
    phonenumber
   FROM buyers;

ALTER TABLE public.vallbuyers
    OWNER TO fer;
```

5. Enseña todos los agentes

```
CREATE OR REPLACE VIEW public.vallagents
 AS
 SELECT identitynumber,
    name,
    address,
    phonenumber,
    officephone
   FROM agents;

ALTER TABLE public.vallagents
    OWNER TO fer;
```

6. Enseña todos los venderdores

```
CREATE OR REPLACE VIEW public.vallsellers
 AS
 SELECT identitynumber,
    name,
    address,
    phonenumber
```

```
  FROM sellers;

ALTER TABLE public.vallsellers
    OWNER TO fer;
```

7. Enseña la bitácora

```
CREATE OR REPLACE VIEW public.vallbinnacle
 AS
 SELECT bi.id_bitacora,
    bi.action,
    bi.username,
    to_char(bi.date::timestamp with time zone, 'YYYY-MM-DD'::text) AS date,
    bi."time"
   FROM bitacora as bi;

ALTER TABLE public.vallbinnacle
    OWNER TO fer;
```

3.1.5. Funciones

1. Enseña las propiedades en el mercado que tiene un agente

```
CREATE OR REPLACE FUNCTION public.fPropMarketPerAgent(
        userid character)
    RETURNS TABLE(propertyid integer, name character varying, address character varying, city character varying, phonenumber integer, bedroomcount integer, features character varying, price real, publicationdate text, image character varying, agentname character varying, agentNumber integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
```

```
    RETURN QUERY
    SELECT
        pom.propertyid,
    pom.name,
    pom.address,
    pom.city,
    pom.phonenumber,
    pom.bedroomcount,
    pom.features,
    pom.price,
    to_char(pom.publicationdate::timestamp with time zone, 'YYYY-MM-DD'::text) AS
publicationdate,
    pom.image,
        ag.name as agentname,
        ag.OfficePhone as agentNumber
   FROM PropertiesOnMarket pom
   INNER JOIN agents ag
   ON pom.AgentIdentityNumber = ag.IdentityNumber
    WHERE pom.AgentIdentityNumber = userid;
END;
$BODY$;

ALTER FUNCTION public.fPropMarketPerAgent(character)
    OWNER TO fer;
```

2. Enseña las propiedades en el mercado que tiene un vendedor

```
CREATE OR REPLACE FUNCTION public.fPropMarketPerSeller(
        userid character)
    RETURNS TABLE(propertyid integer, name character varying, address character varying, city
character varying, phonenumber integer, bedroomcount integer, features character varying, price real,
publicationdate text, image character varying, agentname character varying, agentNumber integer)
    LANGUAGE 'plpgsql'
    COST 100
```

```
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
    RETURN QUERY
    SELECT
            pom.propertyid,
    pom.name,
    pom.address,
    pom.city,
    pom.phonenumber,
    pom.bedroomcount,
    pom.features,
    pom.price,
    to_char(pom.publicationdate::timestamp with time zone, 'YYYY-MM-DD'::text) AS
publicationdate,
    pom.image,
            ag.name as agentname,
            ag.OfficePhone as agentNumber
   FROM PropertiesOnMarket pom
   INNER JOIN agents ag
   ON pom.AgentIdentityNumber = ag.IdentityNumber
    WHERE pom.SellerIdentityNumber = userid;
END;
$BODY$;

ALTER FUNCTION public.fPropMarketPerSeller(character)
    OWNER TO fer;
```

3. Enseña las propiedades vendidas que tiene un agente

```
CREATE OR REPLACE FUNCTION public.fpropsellsperAgent(
```

```
        userid character)
    RETURNS TABLE(propertyid integer, name character varying, address character varying, city character varying, phonenumber integer, bedroomcount integer, features character varying, price real, saleprice real, publicationdate text, saledate text, salecommission real, image character varying)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
    RETURN QUERY
    SELECT
        SoldProperties.propertyid,
    SoldProperties.name,
    SoldProperties.address,
    SoldProperties.city,
    SoldProperties.phonenumber,
    SoldProperties.bedroomcount,
    SoldProperties.features,
    SoldProperties.price,
    SoldProperties.saleprice,
    to_char(SoldProperties.publicationdate::timestamp with time zone, 'YYYY-MM-DD'::text) AS publicationdate,
    to_char(SoldProperties.saledate::timestamp with time zone, 'YYYY-MM-DD'::text) AS saledate,
    SoldProperties.salecommission,
    SoldProperties.image
   FROM SoldProperties
    WHERE SoldProperties.AgentIdentityNumber = userid;
END;
$BODY$;

ALTER FUNCTION public.fpropsellsperAgent(character)
```

```
    OWNER TO fer;
```

4. Enseña las propiedades vendidas que tiene un vendedor

```
CREATE OR REPLACE FUNCTION public.fpropsellsperseller(
        userid character)
    RETURNS TABLE(propertyid integer, name character varying, address character varying, city character varying, phonenumber integer, bedroomcount integer, features character varying, price real, saleprice real, publicationdate text, saledate text, salecommission real, image character varying)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
BEGIN
    RETURN QUERY
    SELECT
        SoldProperties.propertyid,
    SoldProperties.name,
    SoldProperties.address,
    SoldProperties.city,
    SoldProperties.phonenumber,
    SoldProperties.bedroomcount,
    SoldProperties.features,
    SoldProperties.price,
    SoldProperties.saleprice,
    to_char(SoldProperties.publicationdate::timestamp with time zone, 'YYYY-MM-DD'::text) AS publicationdate,
    to_char(SoldProperties.saledate::timestamp with time zone, 'YYYY-MM-DD'::text) AS saledate,
    SoldProperties.salecommission,
    SoldProperties.image
   FROM SoldProperties
    WHERE SoldProperties.SellerIdentityNumber = userid;
```

```
END;
$BODY$;

ALTER FUNCTION public.fpropsellsperseller(character)
   OWNER TO fer;
```

3.1.6. Triggers

1. Function trigger para el CRUD de agente

```
Create function agentData()returns Trigger
as
$$
Declare
Usuario varchar(20) := current_setting('myapp.username');
Fecha date := current_date;
Tiempo time := current_Time;

begin

if TG_OP= 'INSERT' then
insert into "bitacora"(action, username, date, time) values(('Se creo el agente: '||new.name||' con id '||new.identitynumber),Usuario, Fecha, Tiempo);

elsif TG_OP = 'UPDATE' then
insert into "bitacora"(action, username, date, time) values(('Se modifico el agente: '||old.name||' con id '||old.identitynumber),Usuario, Fecha, Tiempo);

elseif TG_OP = 'DELETE' then
insert into "bitacora"(action, username, date, time) values(('Se elimino el agente: '||old.name||' con id '||old.identitynumber),Usuario, Fecha, Tiempo);
end if;

return new;
```

```
end
$$
Language plpgsql;


Create trigger TagentUpdate AFTER update on agents
for each row
execute procedure agentData();

Create trigger TagentInsert after insert on agents
for each row
execute procedure agentData();

Create trigger TagentDelete after delete on agents
for each row
execute procedure agentData();
```

2. Function trigger para el CRUD de Propiedades en el Mercado

```
Create function POMData()returns Trigger
as
$$
Declare
Usuario varchar(20) := current_setting('myapp.username');
Fecha date := current_date;
Tiempo time := current_Time;

begin

if TG_OP= 'INSERT' then
insert into "bitacora"(action, username, date, time) values(('Se creo una nueva propiedad en mercado con id: '||new.propertyid),Usuario, Fecha, Tiempo);
```

```
elsif TG_OP = 'UPDATE' then

insert into "bitacora"(action, username, date, time) values(('Se modifico la propiedad(en mercado): '||old.propertyid),Usuario, Fecha, Tiempo);


elseif TG_OP = 'DELETE' then

insert into "bitacora"(action, username, date, time) values(('Se elimino la propiedad(en mercado): '||old.propertyid),Usuario, Fecha, Tiempo);

end if;


return new;

end

$$

Language plpgsql;



Create trigger TPOMUpdate AFTER update on PropertiesOnMarket

for each row

execute procedure POMData();


Create trigger TPOMInsert after insert on PropertiesOnMarket

for each row

execute procedure POMData();


Create trigger TPOMDelete after delete on PropertiesOnMarket

for each row

execute procedure POMData();
```

3. Function trigger para el CRUD de Propiedades Vendidas

```
Create function SPData()returns Trigger

as

$$

Declare

Usuario varchar(20) := current_setting('myapp.username');
```

```
Fecha date := current_date;

Tiempo time := current_Time;


begin


if TG_OP= 'INSERT' then

insert into "bitacora"(action, username, date, time) values(('Se creo una nueva propiedad vendida con id: '||new.propertyid),Usuario, Fecha, Tiempo);


elsif TG_OP = 'UPDATE' then

insert into "bitacora"(action, username, date, time) values(('Se modifico la propiedad(vendida): '||old.propertyid),Usuario, Fecha, Tiempo);


elseif TG_OP = 'DELETE' then

insert into "bitacora"(action, username, date, time) values(('Se elimino la propiedad(vendida): '||old.propertyid),Usuario, Fecha, Tiempo);

end if;


return new;

end

$$

Language plpgsql;



Create trigger TSPUpdate AFTER update on SoldProperties

for each row

execute procedure SPData();


Create trigger TSPInsert after insert on SoldProperties

for each row

execute procedure SPData();


Create trigger TSPDelete after delete on SoldProperties
```

```
for each row
execute procedure SPData();
```

4. Function trigger para el CRUD del comprador

```
CREATE FUNCTION buyersData()RETURNS TRIGGER
AS
$$
DECLARE
Usuario varchar(20) := current_setting('myapp.username');
Fecha date := current_date;
Tiempo time := current_Time;

BEGIN

IF TG_OP= 'INSERT' THEN
INSERT INTO "bitacora"(action, username, date, time) VALUES(('Se creo el comprador: '||new.name||' con id '||new.identitynumber),Usuario, Fecha, Tiempo);

ELSIF TG_OP = 'UPDATE' THEN
INSERT INTO "bitacora"(action, username, date, time) VALUES(('Se modifico el comprador: '||old.name||' con id '||old.identitynumber),Usuario, Fecha, Tiempo);

ELSEIF TG_OP = 'DELETE' THEN
INSERT INTO "bitacora"(action, username, date, time) VALUES(('Se elimino el comprador: '||old.name||' con id '||old.identitynumber),Usuario, Fecha, Tiempo);
END IF;

RETURN NEW;
END
$$
LANGUAGE plpgsql;

CREATE TRIGGER tbuyerInsert AFTER INSERT ON buyers
```

```
FOR EACH ROW
EXECUTE PROCEDURE buyersData();

CREATE TRIGGER tbuyerUpdate AFTER UPDATE ON buyers
FOR EACH ROW
EXECUTE PROCEDURE buyersData();

CREATE TRIGGER tbuyerDelete AFTER DELETE ON buyers
FOR EACH ROW
EXECUTE PROCEDURE buyersData();
```

5. Function trigger para el CRUD de vendedores

```
CREATE FUNCTION sellersData()RETURNS TRIGGER
AS
$$
DECLARE
Usuario varchar(20) := current_setting('myapp.username');
Fecha date := current_date;
Tiempo time := current_Time;

BEGIN

IF TG_OP= 'INSERT' THEN
INSERT INTO "bitacora"(action, username, date, time) VALUES(('Se creo el vendedor: '||new.name||' con id '||new.identitynumber),Usuario, Fecha, Tiempo);

ELSIF TG_OP = 'UPDATE' THEN
INSERT INTO "bitacora"(action, username, date, time) VALUES(('Se modifico el vendedor: '||old.name||' con id '||old.identitynumber),Usuario, Fecha, Tiempo);

ELSEIF TG_OP = 'DELETE' THEN
INSERT INTO "bitacora"(action, username, date, time) VALUES(('Se elimino el vendedor: '||old.name||' con id '||old.identitynumber),Usuario, Fecha, Tiempo);
```

```
END IF;

RETURN NEW;
END
$$
LANGUAGE plpgsql;

CREATE TRIGGER tsellerInsert AFTER INSERT ON sellers
FOR EACH ROW
EXECUTE PROCEDURE sellersData();

CREATE TRIGGER tsellerUpdate AFTER UPDATE ON sellers
FOR EACH ROW
EXECUTE PROCEDURE sellersData();

CREATE TRIGGER tsellerDelete AFTER DELETE ON sellers
FOR EACH ROW
EXECUTE PROCEDURE sellersData();
```

4. Anexos

4.1. Manual del Usuario

Requerimientos:

1. Clonar el repositorio de Github (https://github.com/fercast711/final_project_db1.git)
2. Abrir el código con Visual Studio Code
3. Verificar la conexión a la base de datos local y la base datos en AWS en el .env
4. Ctrl+J para abrir una terminal y levantar el Backend con los comandos: cd backend -> npm i - > npm run dev
5. Ctrl+J para abrir una nueva terminal y levantar el Frontend con los comandos: cd frontend -> npm i - > npm run dev
6. Ingresar al link generado al levantar el Frontend (Debería ser similar a este: http://localhost:5173/)

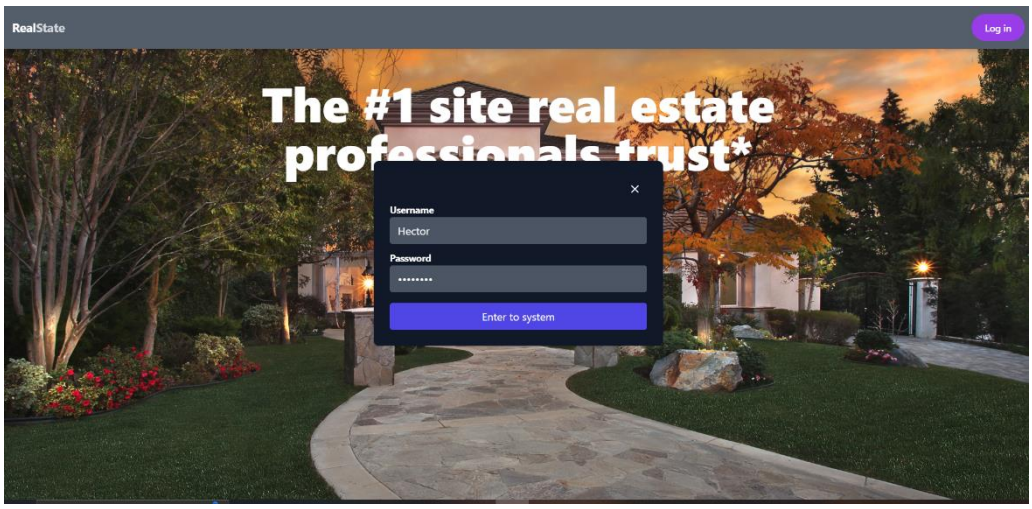
4.1.1. Rol de Administrador

Login Admin:

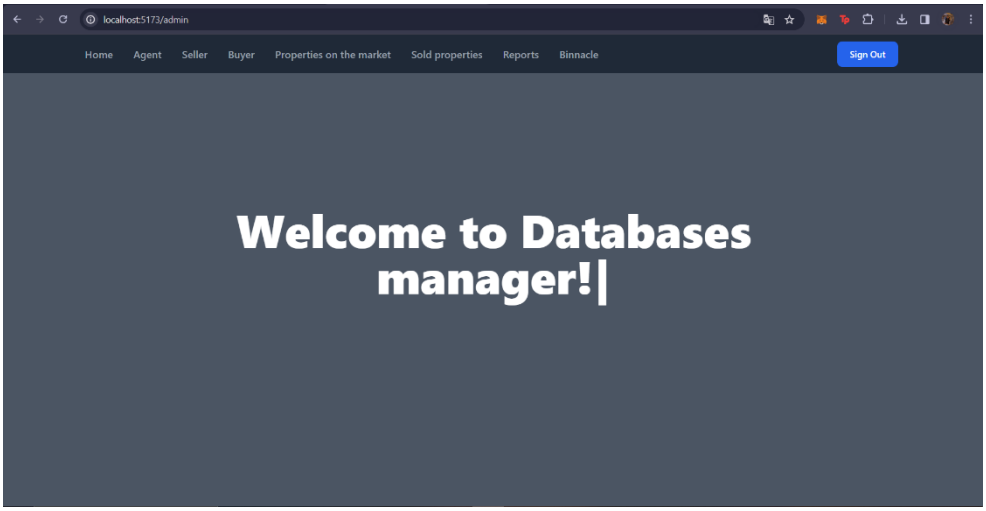
Ingresar al link: http://localhost:5173/

Pasos:

1. Click en el botón “Log In”
2. Ingresar su nombre usuario, en este caso el de los administradores (Hector, fercast, Tatiana o Andrea)
3. Ingresar la contraseña “12345678”
4. Click en el botón “Enter to System”

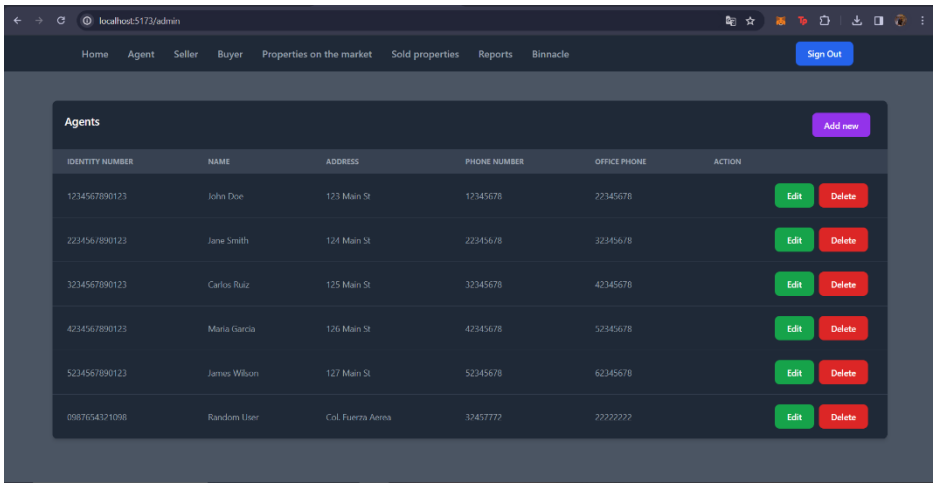


Pantalla Home:



CRUD Agents:

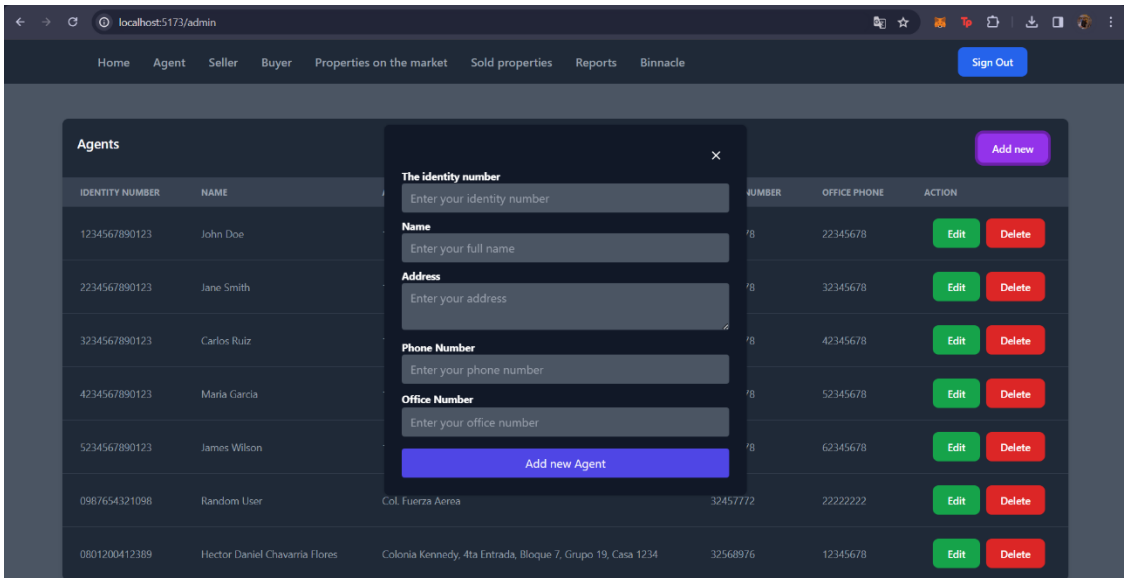
Click en el botón “Agent”



Ingresar un nuevo agente:

Pasos:

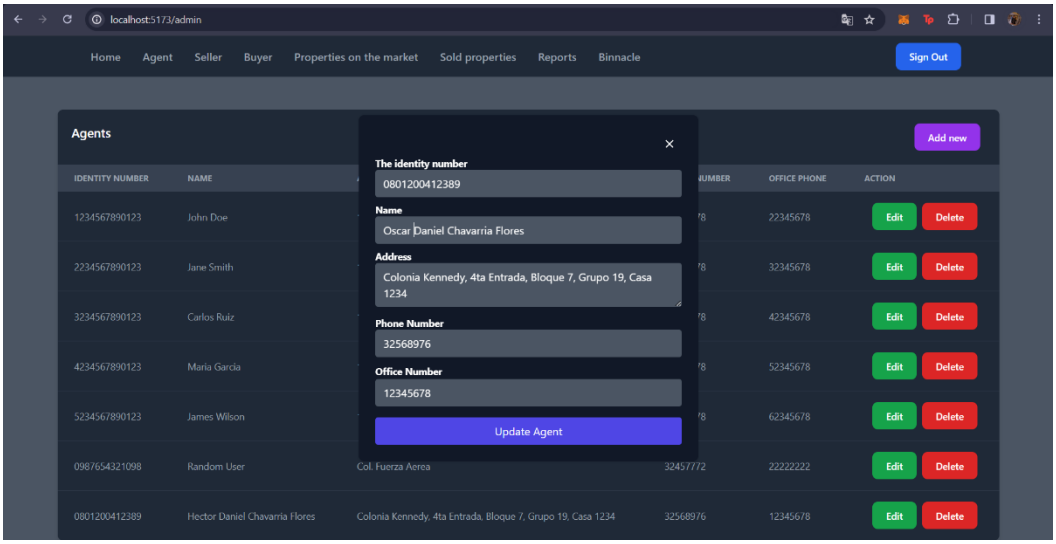
1. Click en el botón morado “Add new”
2. Ingresar el número de identidad del agente
3. Ingresar el nombre completo del agente
4. Ingresar la dirección detallada del agente
5. Ingresar el número de teléfono del agente
6. Ingresar el número de oficina del agente
7. Click en el botón azul “Add new Agent”



Editar la información de un agente:

Pasos:

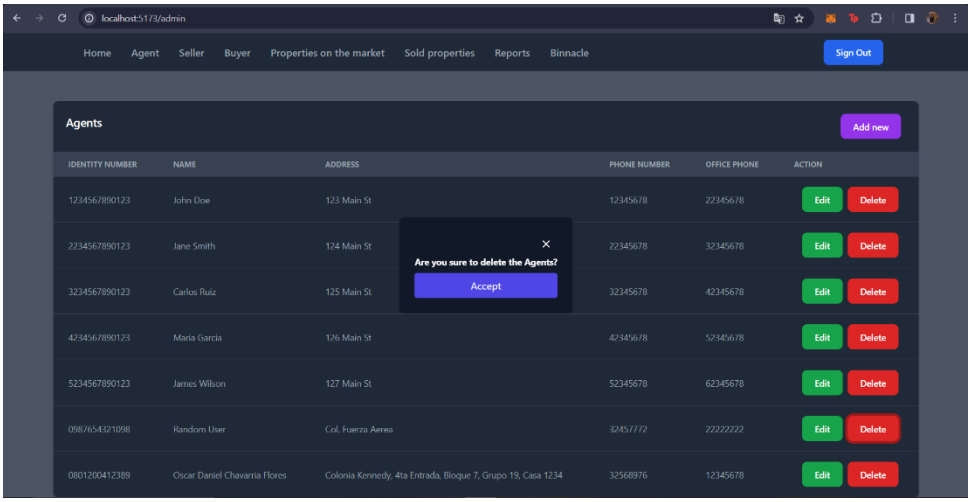
1. Click en el botón verde “Edit”
2. Editar la información necesaria
3. Click en el botón azul “Update Agent”



Eliminar un agente:

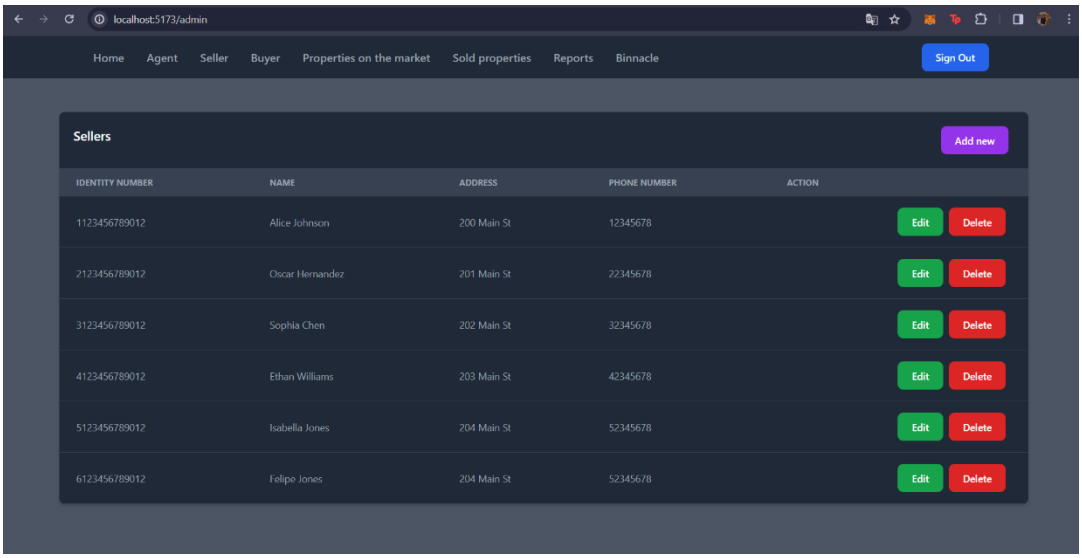
Pasos:

1. Click en el botón rojo “Delete”
2. Click en el botón azul “Accept” si está de acuerdo en que se elimine ese agente



CRUD Seller:

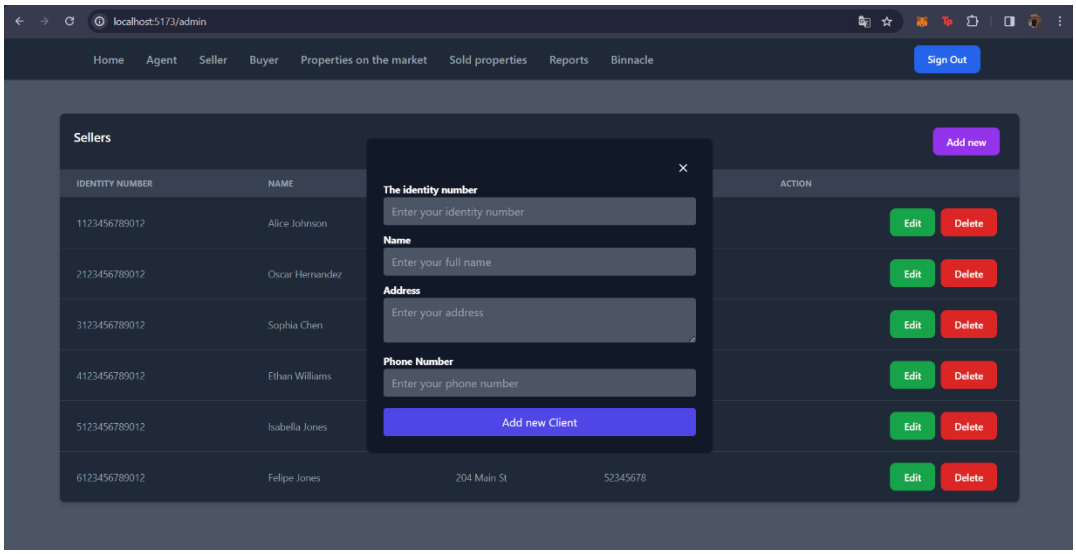
Click en el botón “Seller”



Ingresar un nuevo vendedor:

Pasos:

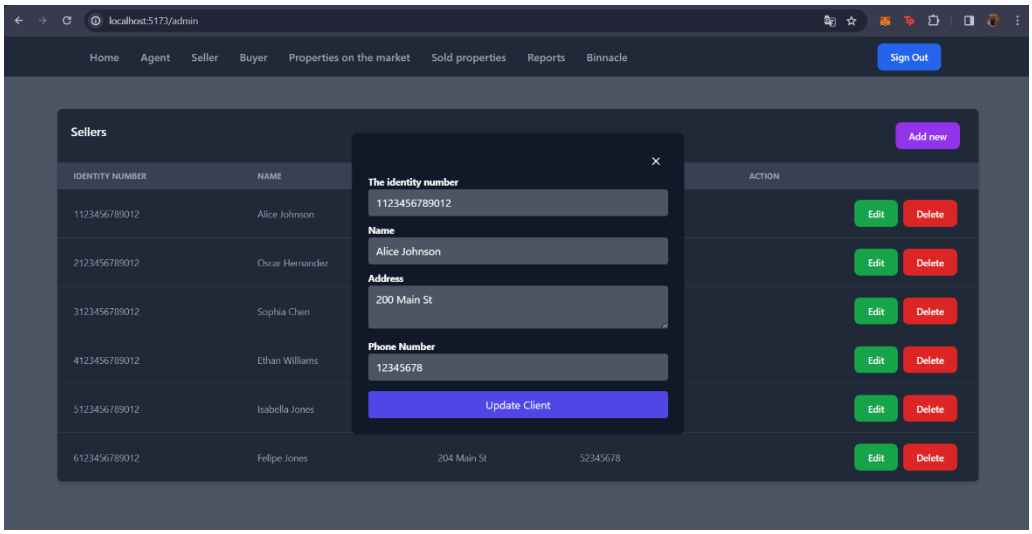
1. Click en el botón morado “Add new”
2. Ingresar el número de identidad del vendedor
3. Ingresar el nombre completo del vendedor
4. Ingresar la dirección detallada del vendedor
5. Ingresar el número de teléfono del vendedor
6. Click en el botón azul “Add new Client”



Editar la información de un vendedor:

Pasos:

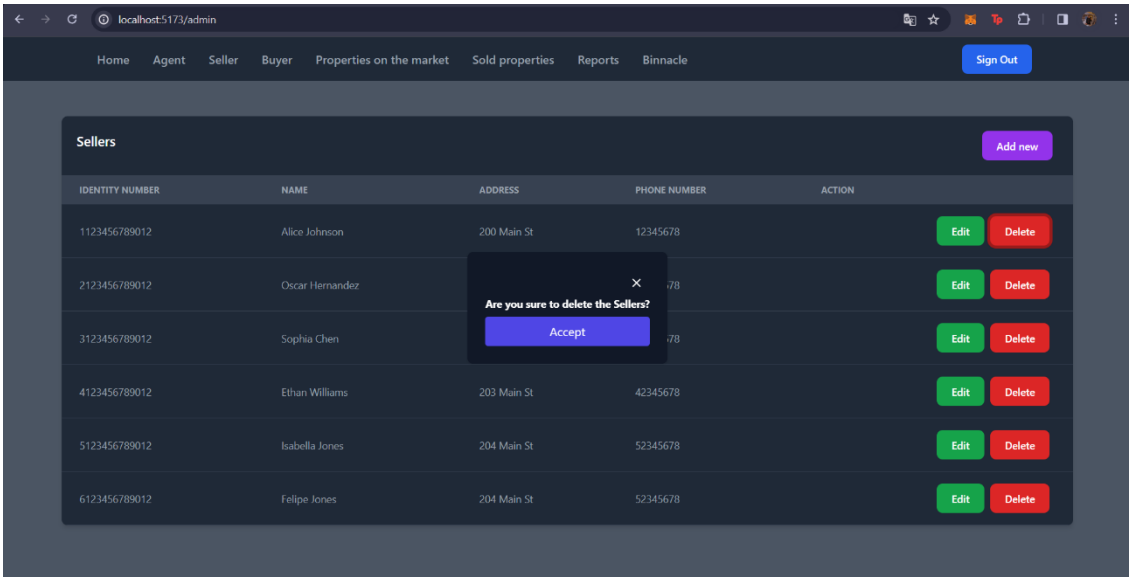
1. Click en el botón verde “Edit”
2. Editar la información necesaria
3. Click en el botón azul “Update Client”



Eliminar un vendedor:

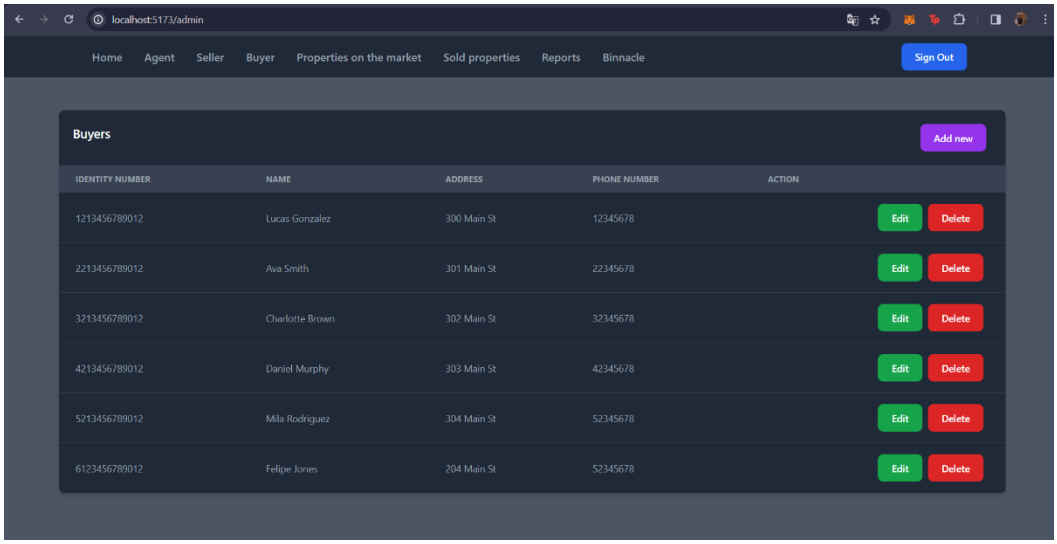
Pasos:

1. Click en el botón rojo “Delete”
2. Click en el botón azul “Accept” si está de acuerdo en que se elimine ese vendedor



CRUD Buyer:

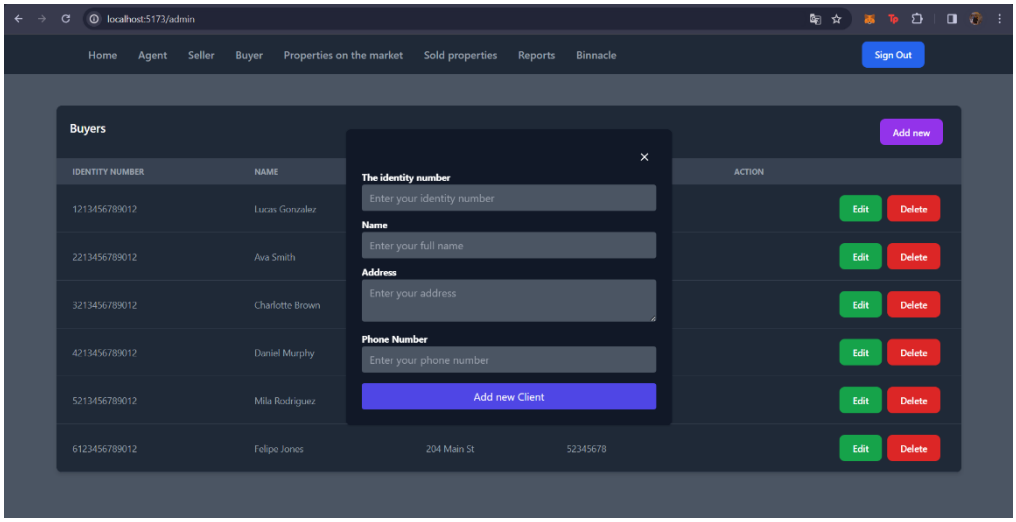
Click en el botón “Buyer”



Ingresar un nuevo comprador:

Pasos:

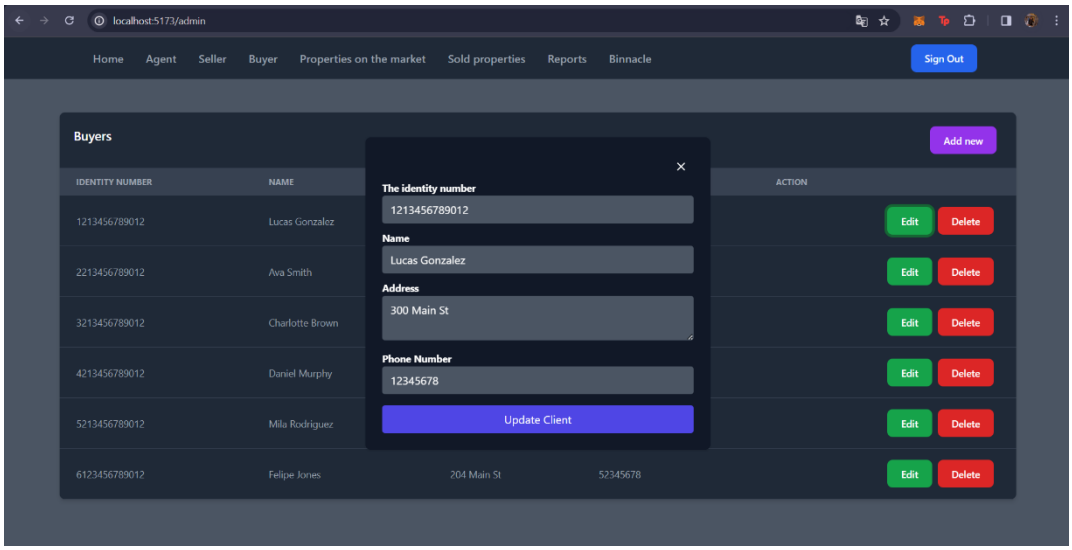
1. Click en el botón morado “Add new”
2. Ingresar el número de identidad del comprador
3. Ingresar el nombre completo del comprador
4. Ingresar la dirección detallada del comprador
5. Ingresar el número de teléfono del comprador
6. Click en el botón azul “Add new Client”



Editar la información de un comprador:

Pasos:

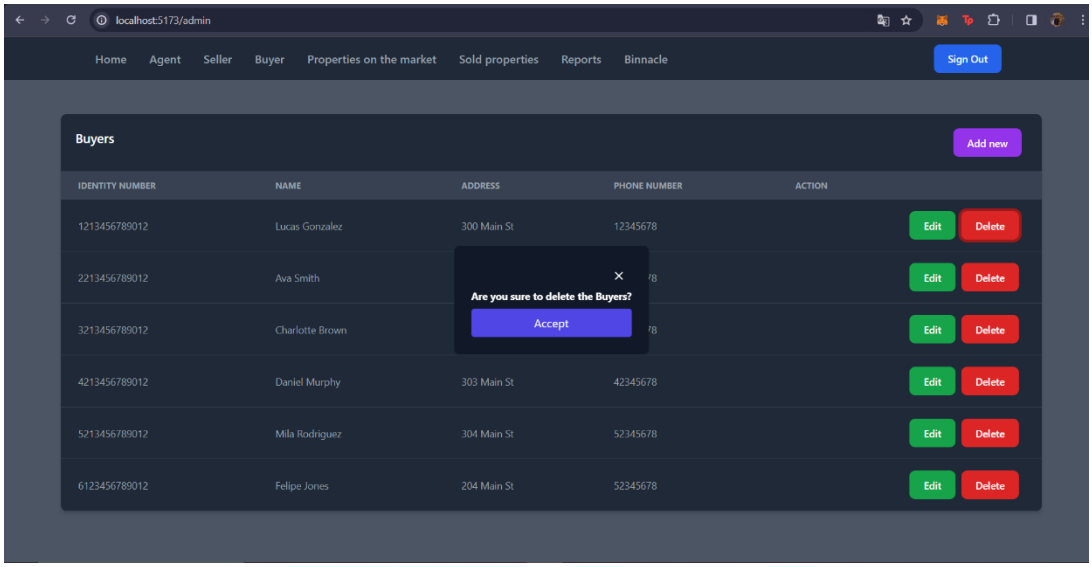
1. Click en el botón verde “Edit”
2. Editar la información necesaria
3. Click en el botón azul “Update Client”



Eliminar un comprador:

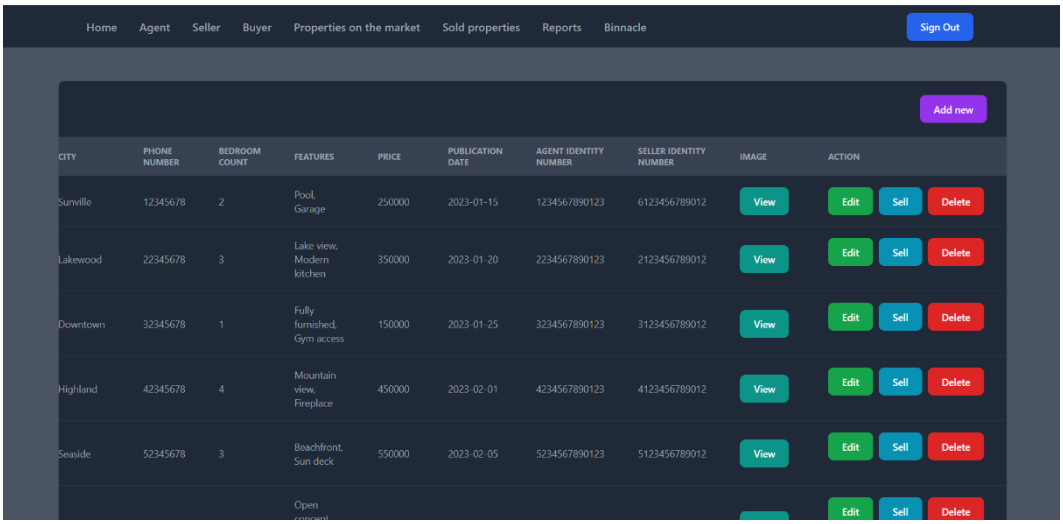
Pasos:

1. Click en el botón rojo “Delete”
2. Click en el botón azul “Accept” si está de acuerdo en que se elimine ese comprador



CRUD Propiedades en el Mercado:

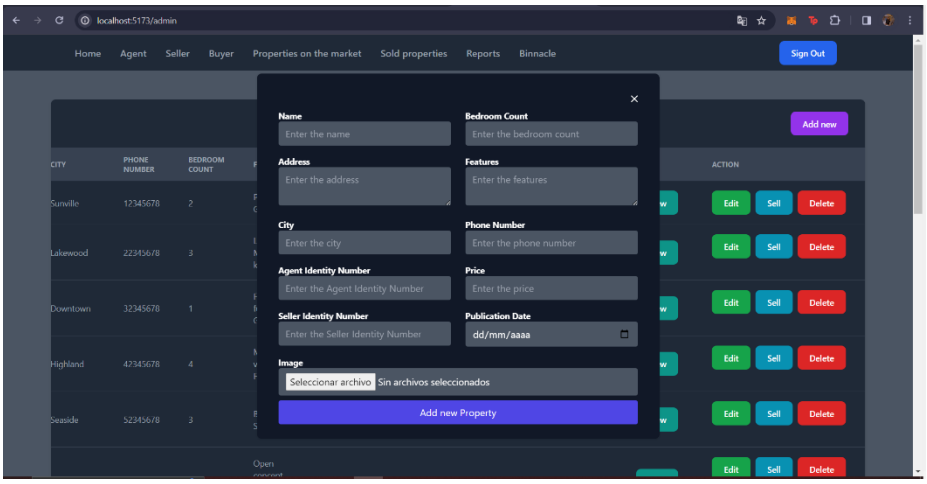
Click en el botón “Properties on the market”



Ingresar una nueva propiedad al mercado:

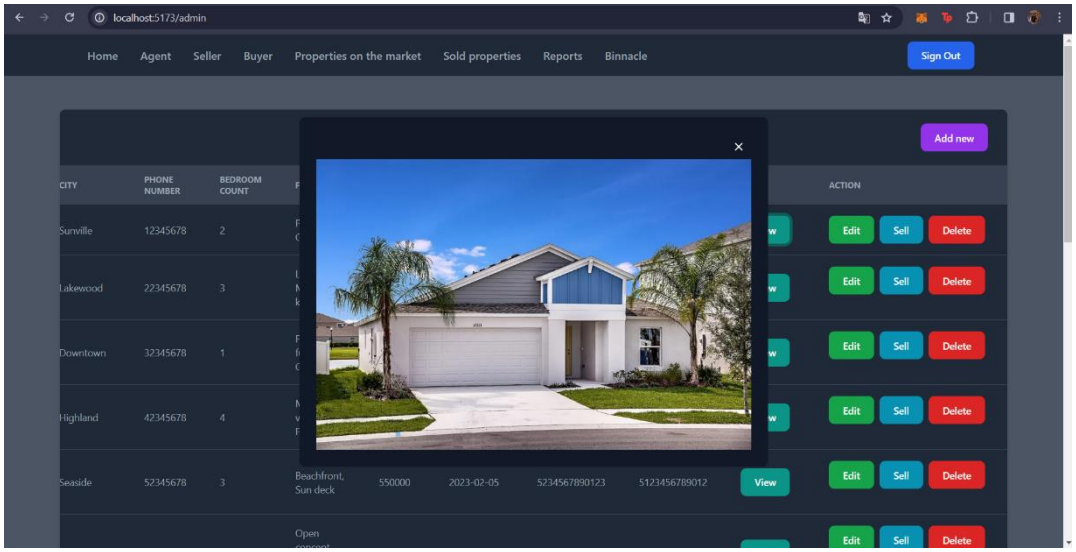
Pasos:

1. Click en el botón morado “Add new”
2. Ingresar el nombre completo de la propiedad
3. Ingresar el número de habitaciones de la propiedad
4. Ingresar la dirección detallada de la propiedad
5. Ingresar las características de la propiedad
6. Ingresar la ciudad donde se encuentra la propiedad
7. Ingresar el número de teléfono de la propiedad
8. Ingresar el número de identidad del agente que lo atendió
9. Ingresar el precio de la propiedad
10. Ingresar el número de identidad del vendedor de la propiedad
11. Ingresar la fecha de publicación
12. Adjuntar una imagen de la propiedad
13. Click en el botón azul “Add new Property”



Visualizar la propiedad en el mercado:

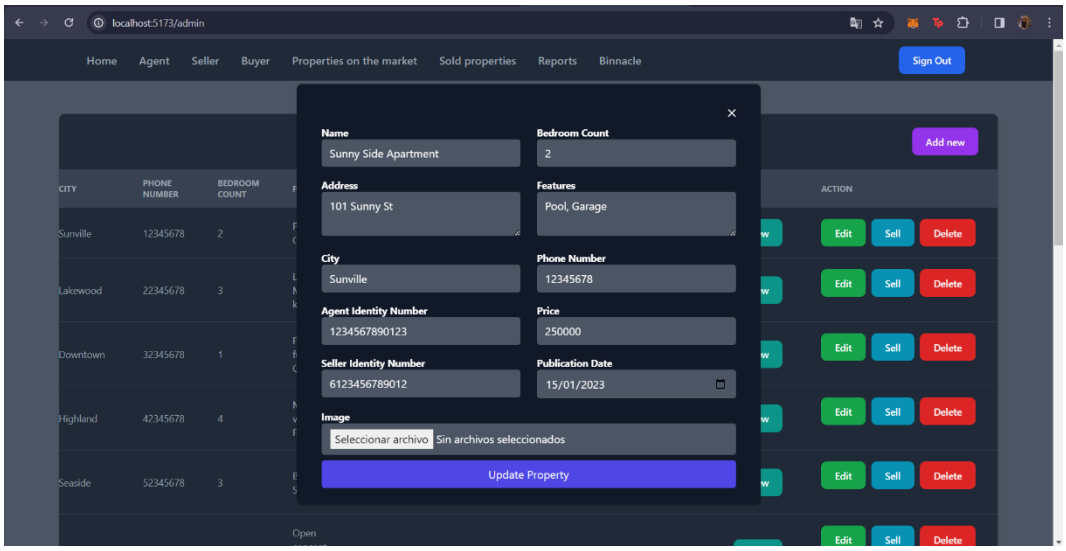
Click en el botón “View”



Editar la información de una propiedad en el mercado:

Pasos:

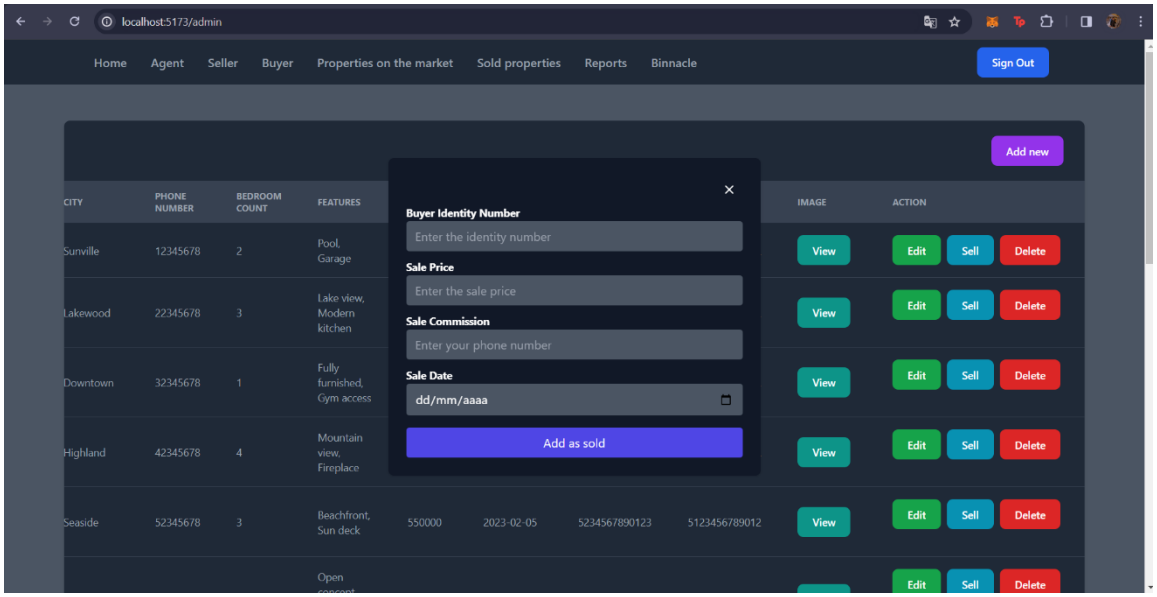
1. Click en el botón verde “Edit”
2. Editar la información necesaria
3. Click en el botón azul “Update Property”



Vender una propiedad:

Pasos:

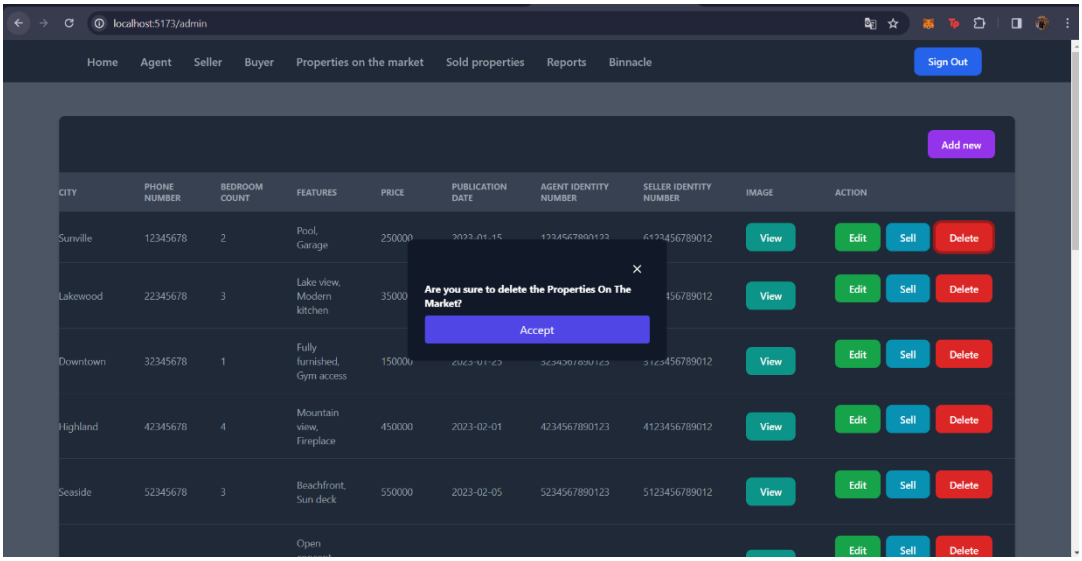
1. Click en el botón azul “Sell”
2. Ingresar el número de identidad del comprador
3. Ingresar el número de identidad del vendedor
4. Ingresar la comisión por la venta
5. Seleccionar la fecha de venta
6. Click en el botón “Add as sold”



Eliminar una propiedad en el mercado:

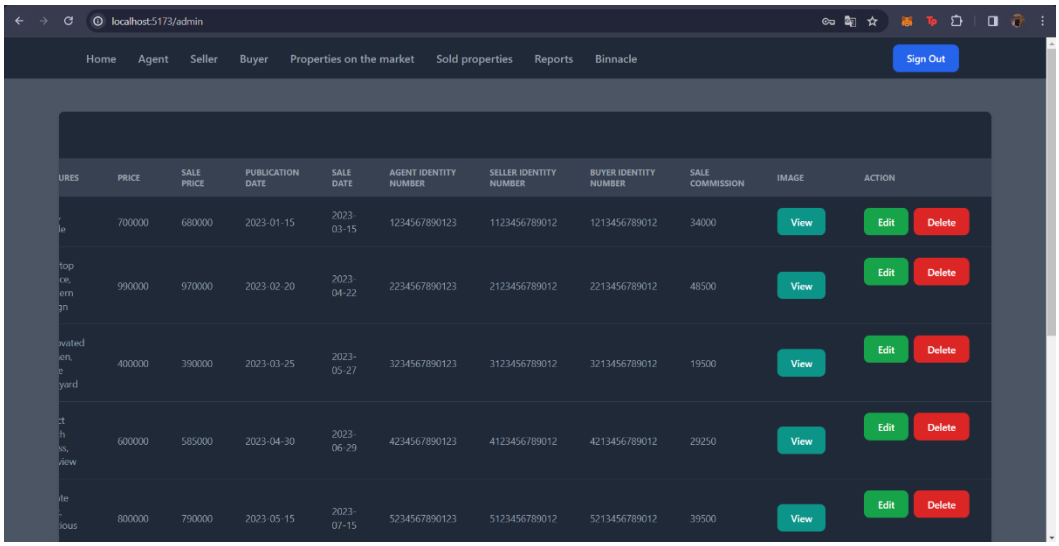
Pasos:

1. Click en el botón rojo “Delete”
2. Click en el botón azul “Accept” si está de acuerdo en que se elimine esa propiedad



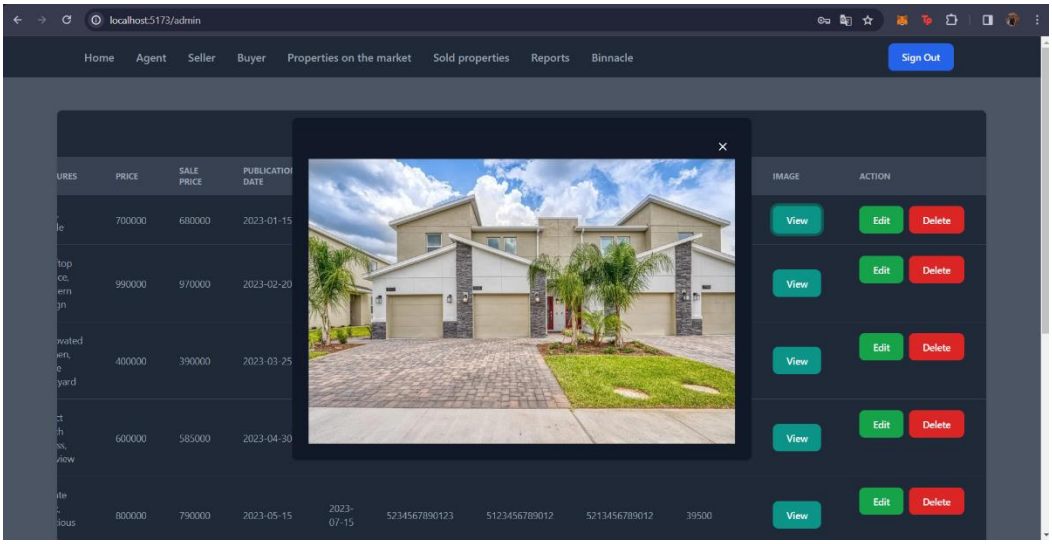
CRUD Propiedades Vendidas:

Click en el botón “Sold Properties”



Visualizar la propiedad vendida:

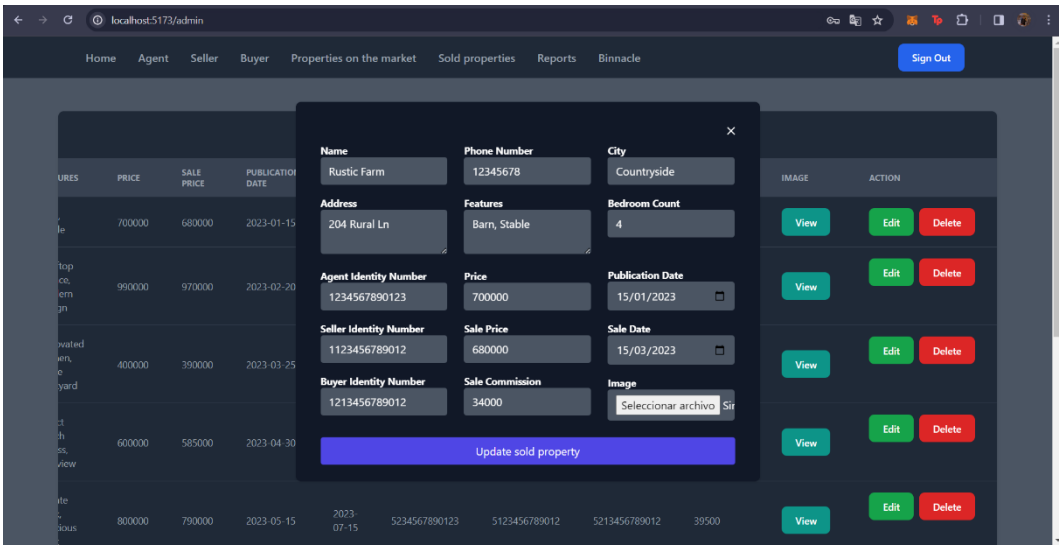
Click en el botón “View”



Editar la información de una propiedad vendida:

Pasos:

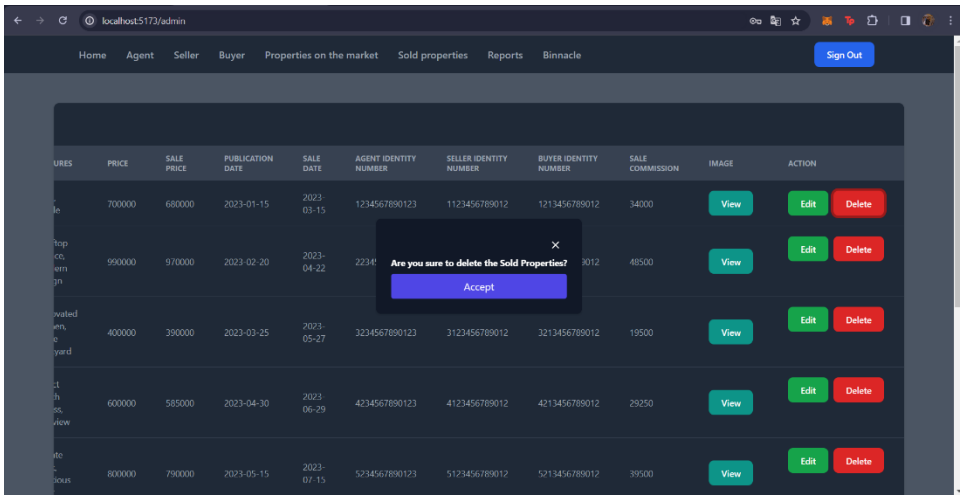
1. Click en el botón verde “Edit”
2. Editar la información necesaria
3. Click en el botón azul “Update sold property”



Eliminar una propiedad vendida:

Pasos:

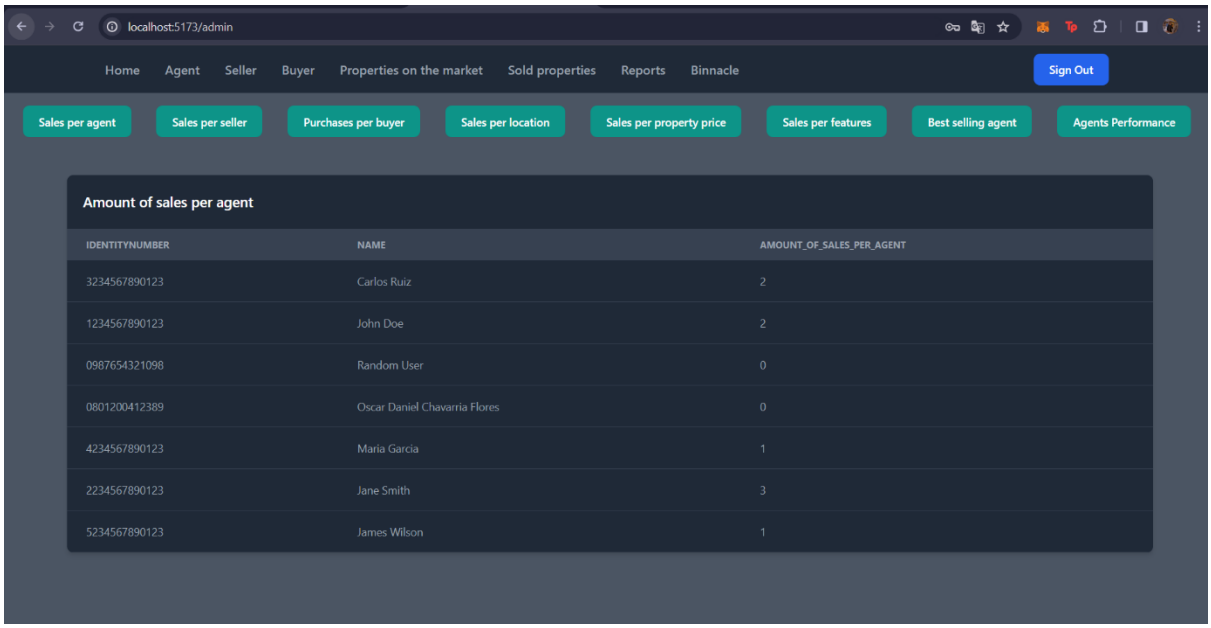
1. Click en el botón rojo “Delete”
2. Click en el botón azul “Accept” si está de acuerdo en que se elimine esa propiedad vendida



Reportes

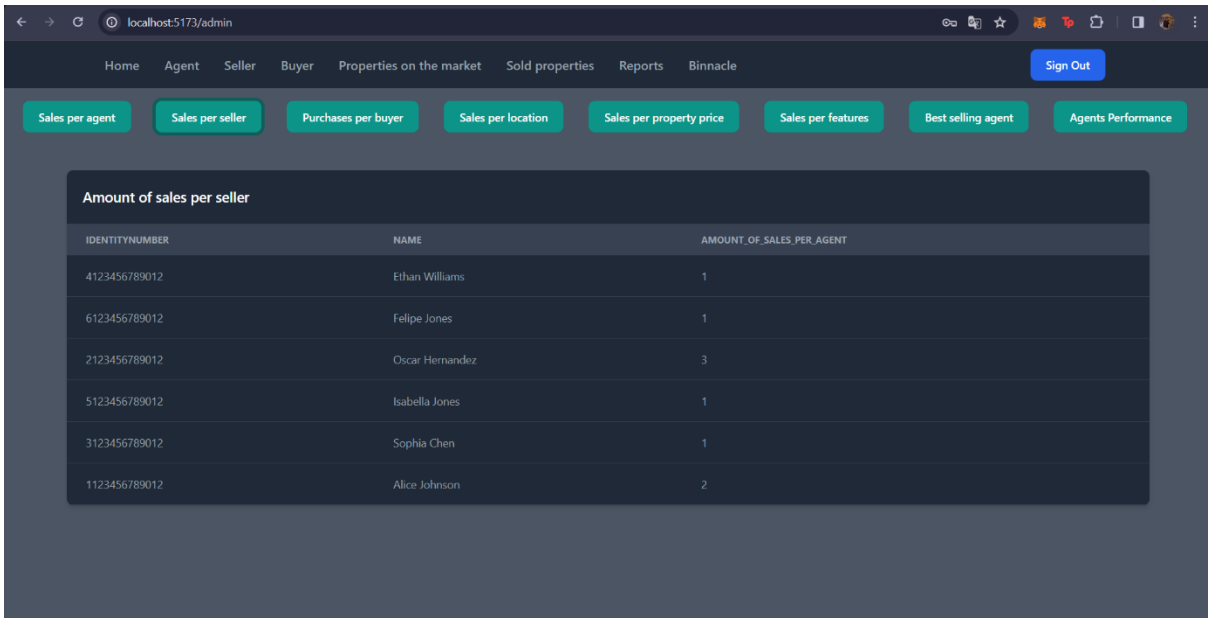
Ventas por agente

Click en el botón verde “Sales per agent”



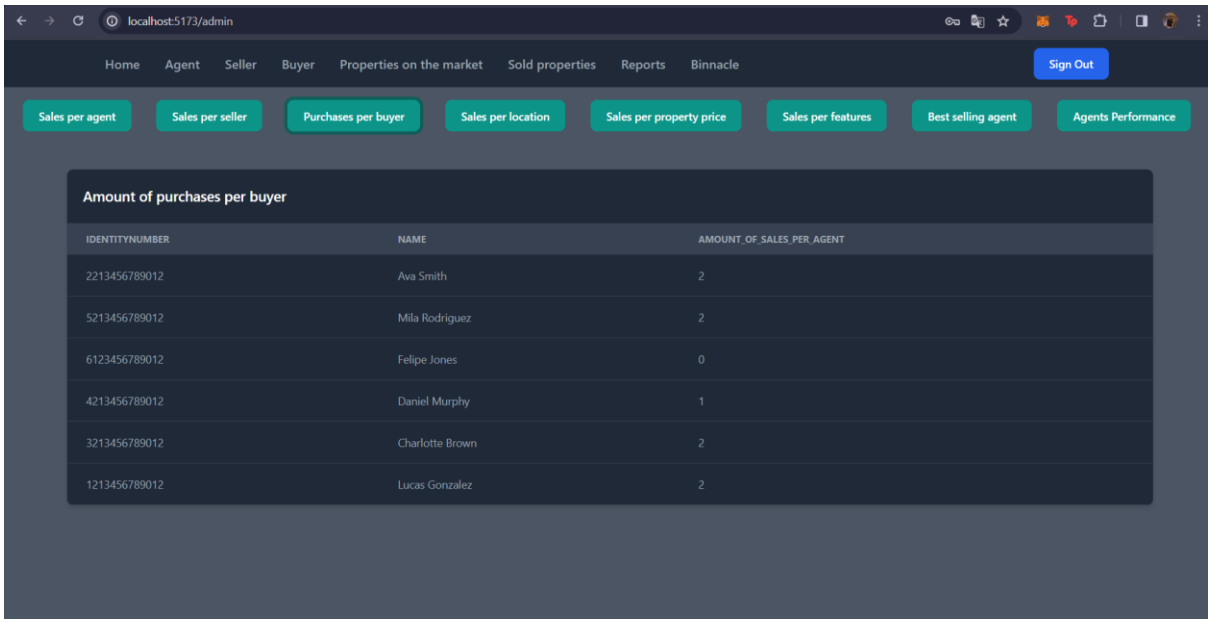
Ventas por vendedor

Click en el botón verde “Sales per seller”



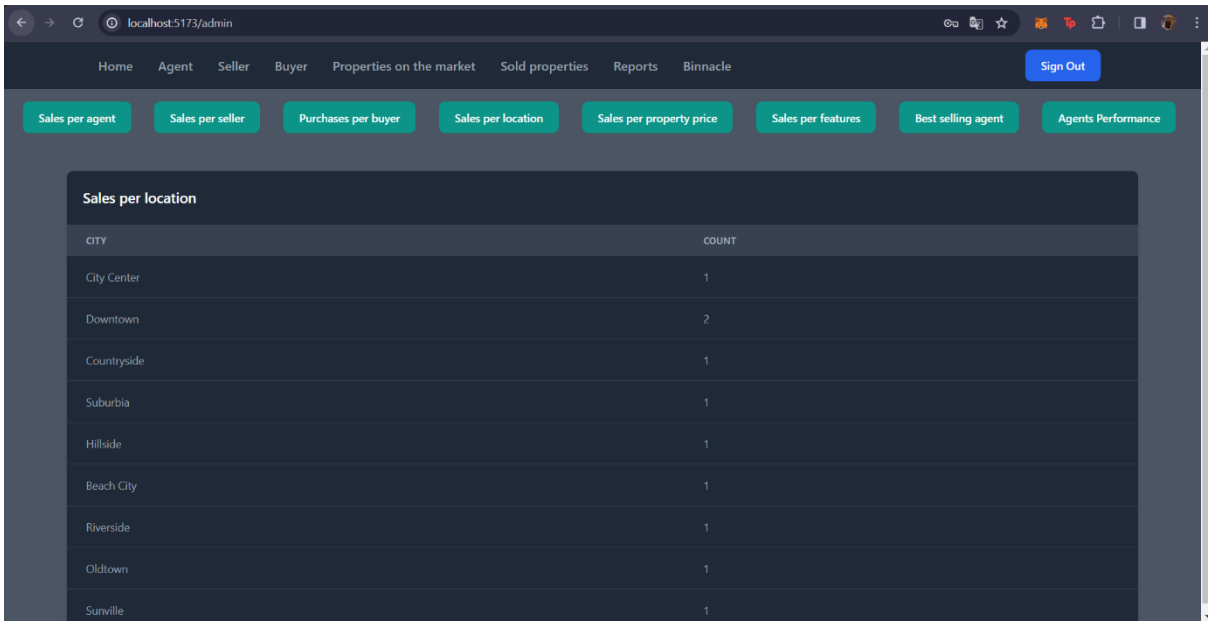
Compras por comprador

Click en el botón verde “Purchases per buyer”



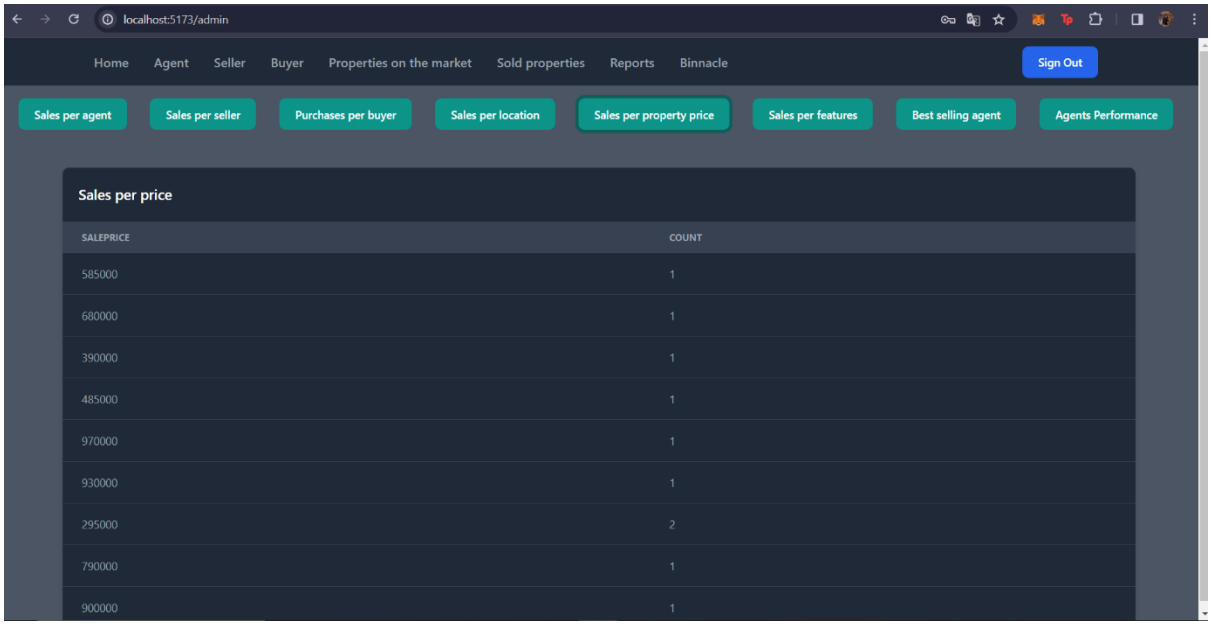
Ventas por ubicación

Click en el botón verde “Sales per location”



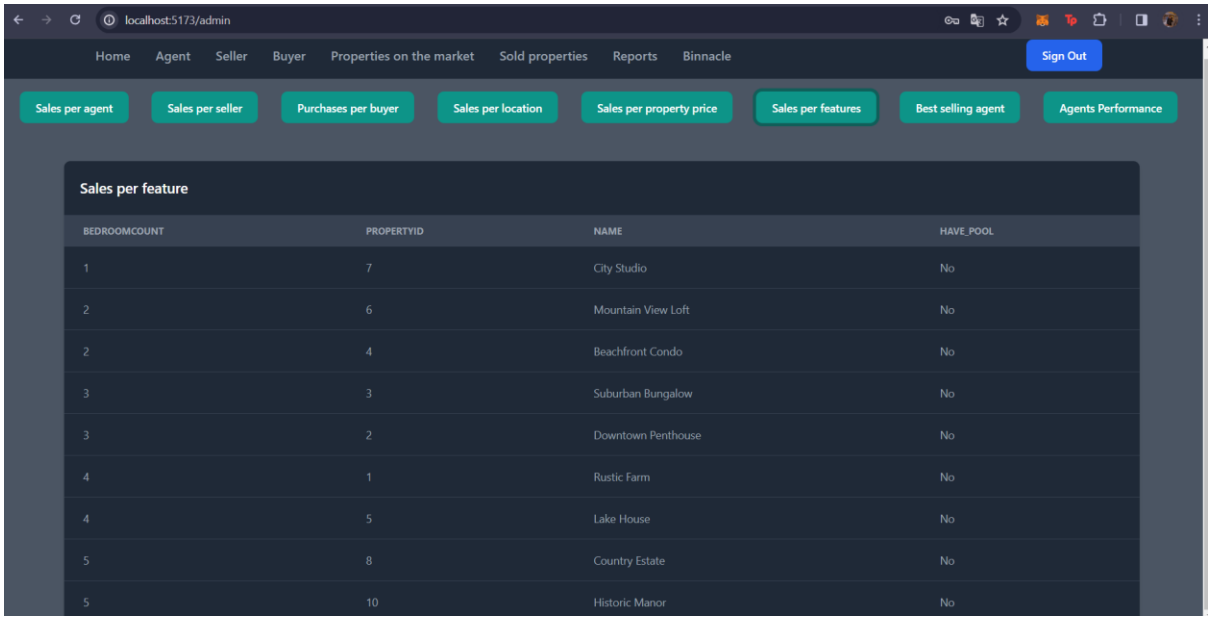
Ventas por precio de propiedad

Click en el botón verde “Sales per property price”



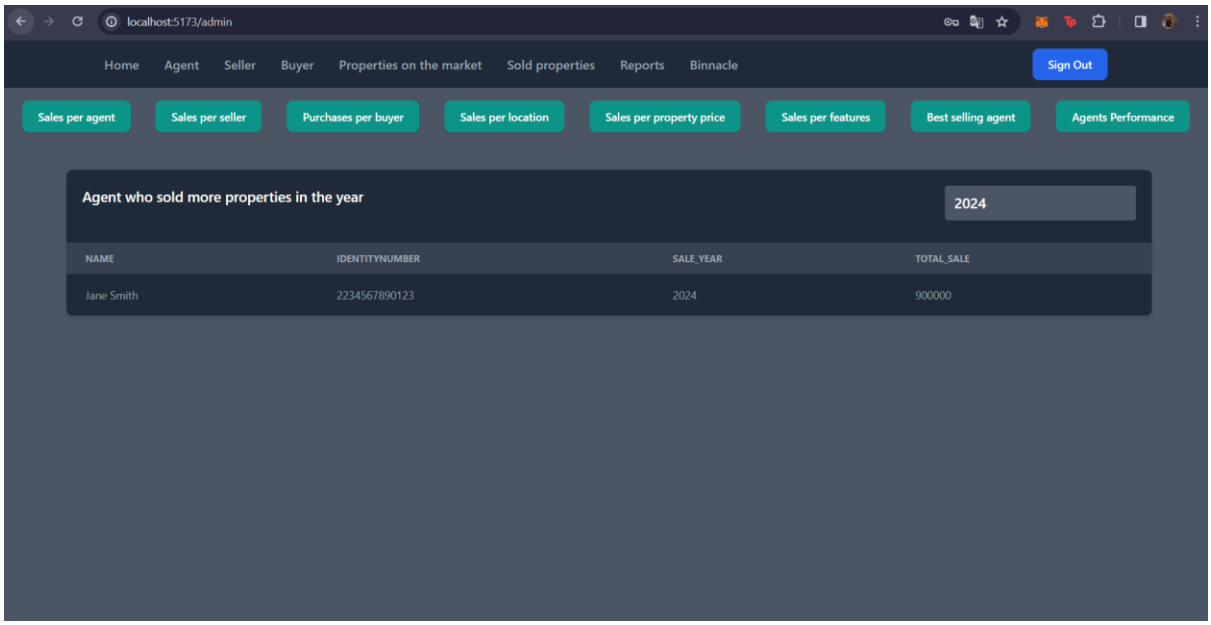
Ventas por características

Click en el botón verde “Sales per features”



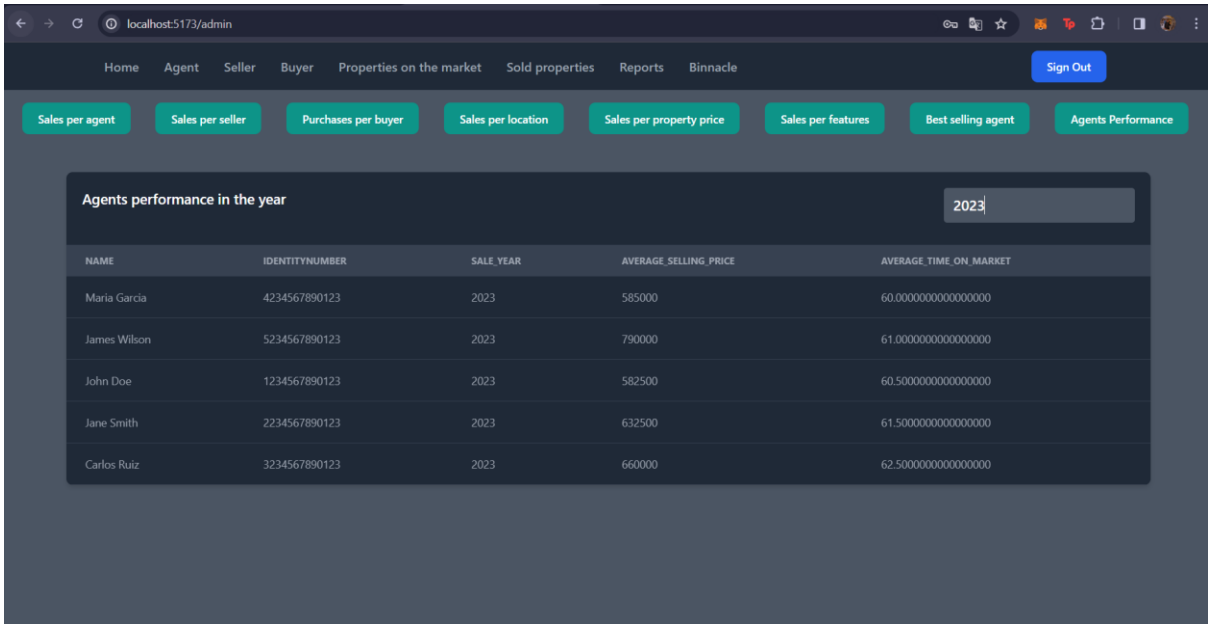
Agente que vendió la mayor cantidad de propiedades en el año por valor total

Click en el botón verde “Best selling agent” e ingresar el año que desea visualizar



Rendimiento de los agentes

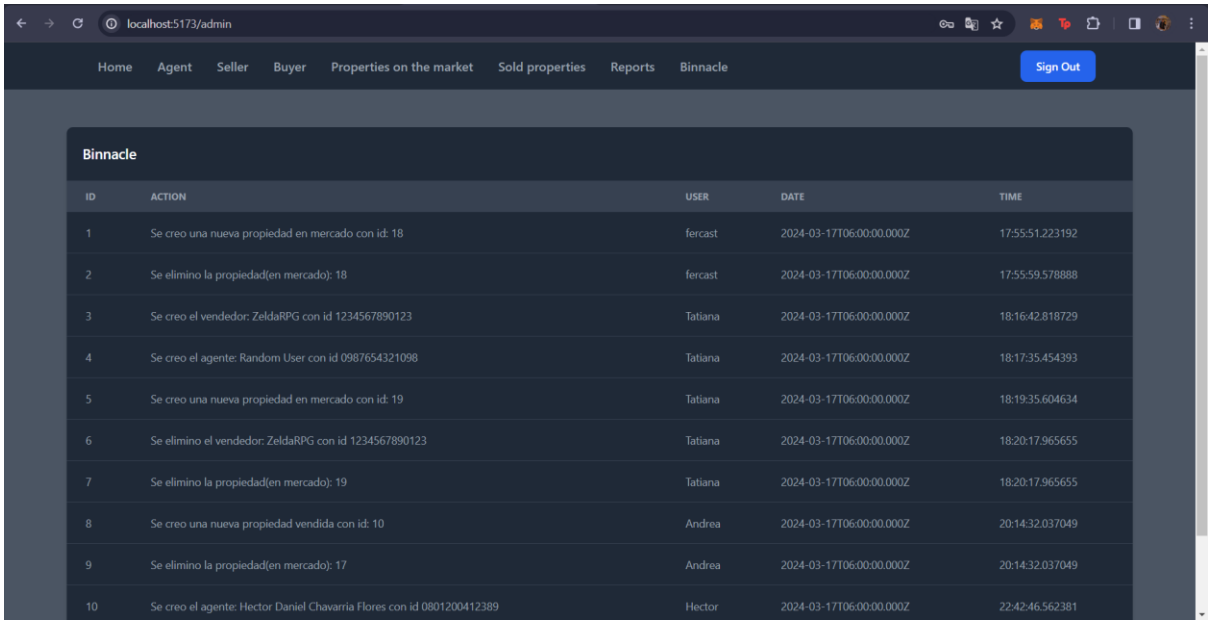
Click en el botón verde “Agents performance” e ingresar el año que desea visualizar



Bitácora

Para visualizar la bitácora de cambios en el sistema:

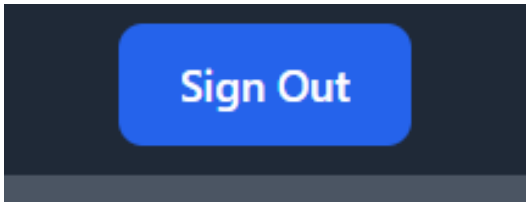
Click en el botón “Binnacle”



Salida del sistema

Para salir del sistema:

Click en el botón azul “Sign Out”

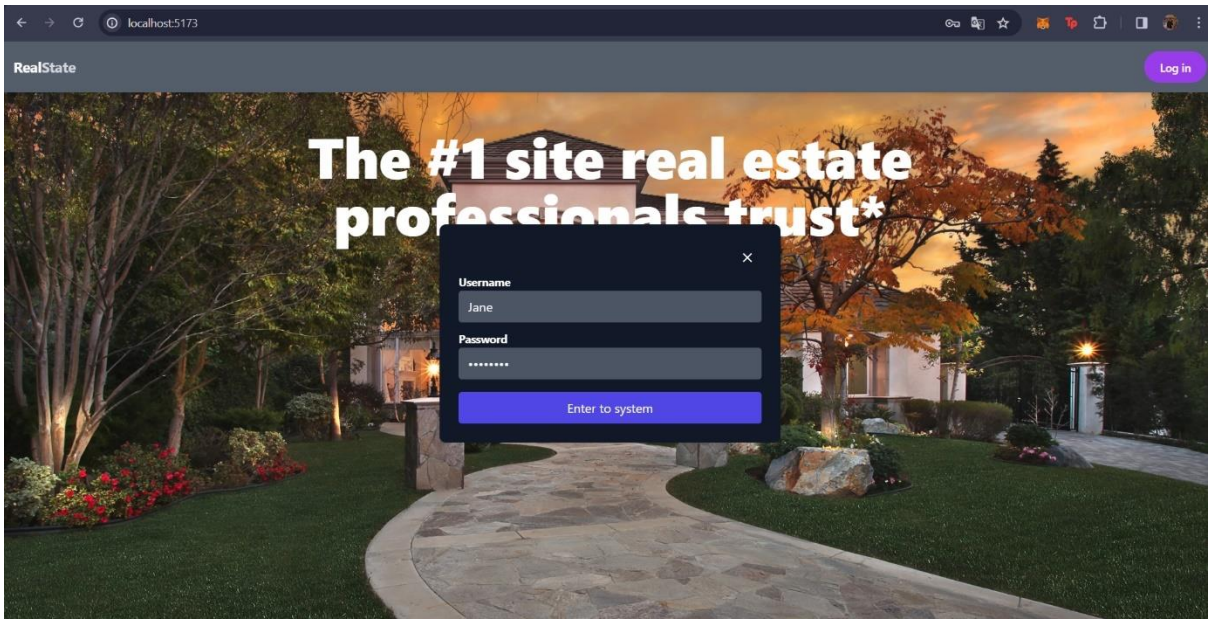


4.1.2. Rol de Agente

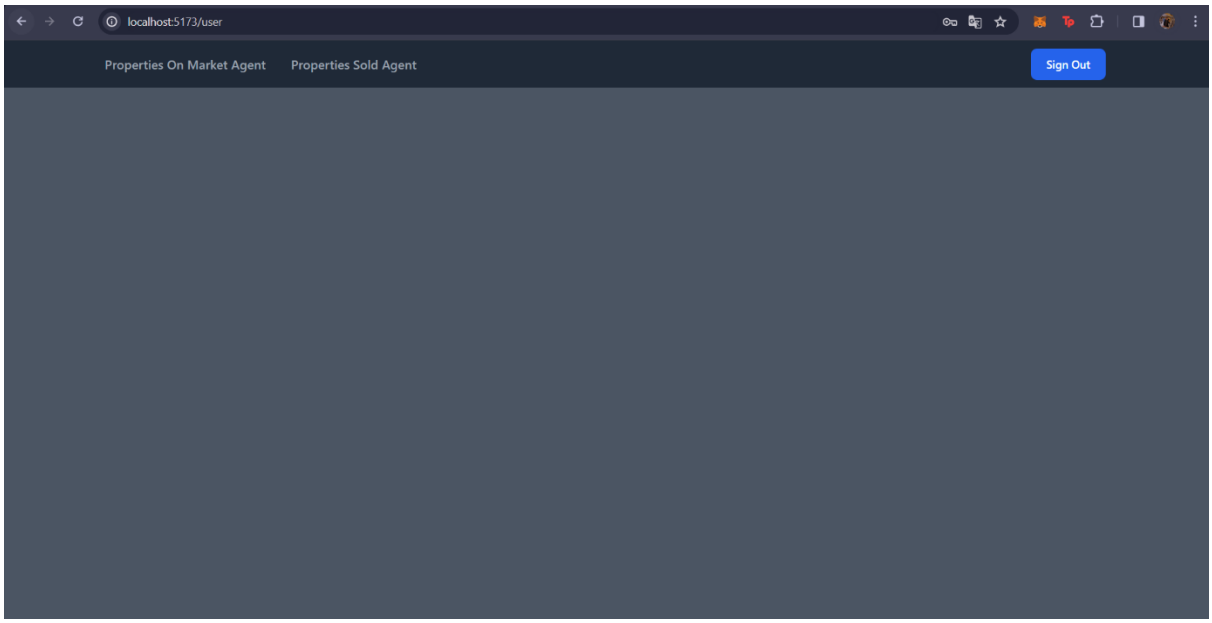
Log In Agente:

Pasos:

1. Click en el botón “Log In”
2. Ingresar su nombre usuario, en este caso el de los agentes (Jane)
3. Ingresar la contraseña “12345678”
4. Click en el botón “Enter to System”

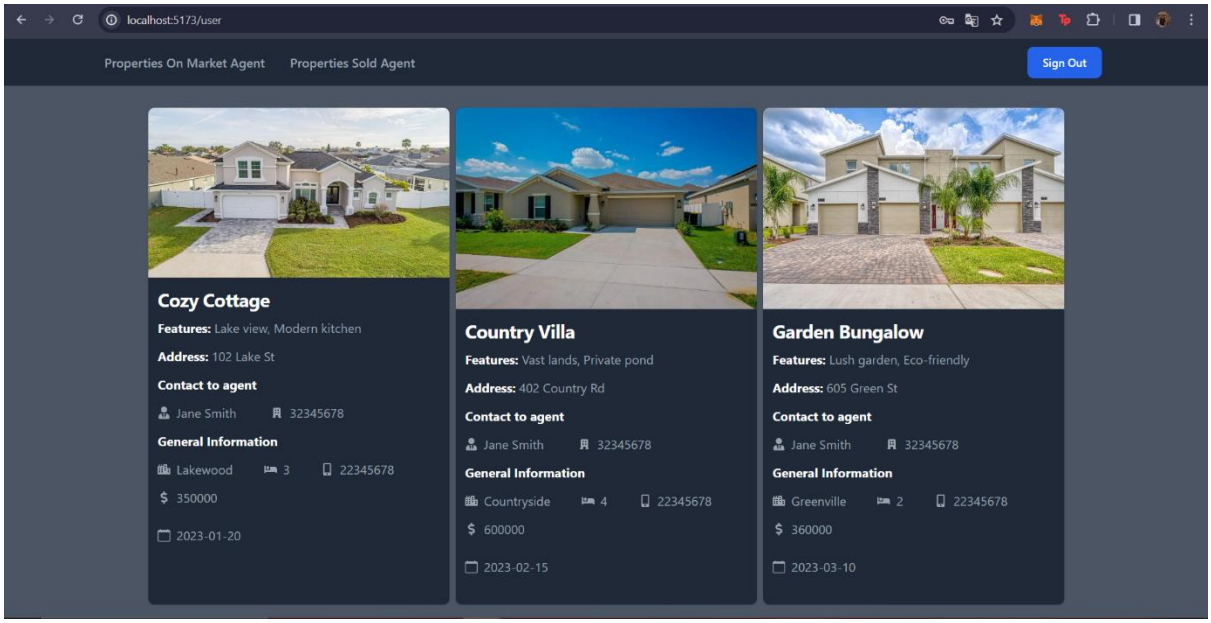


Pantalla Home:



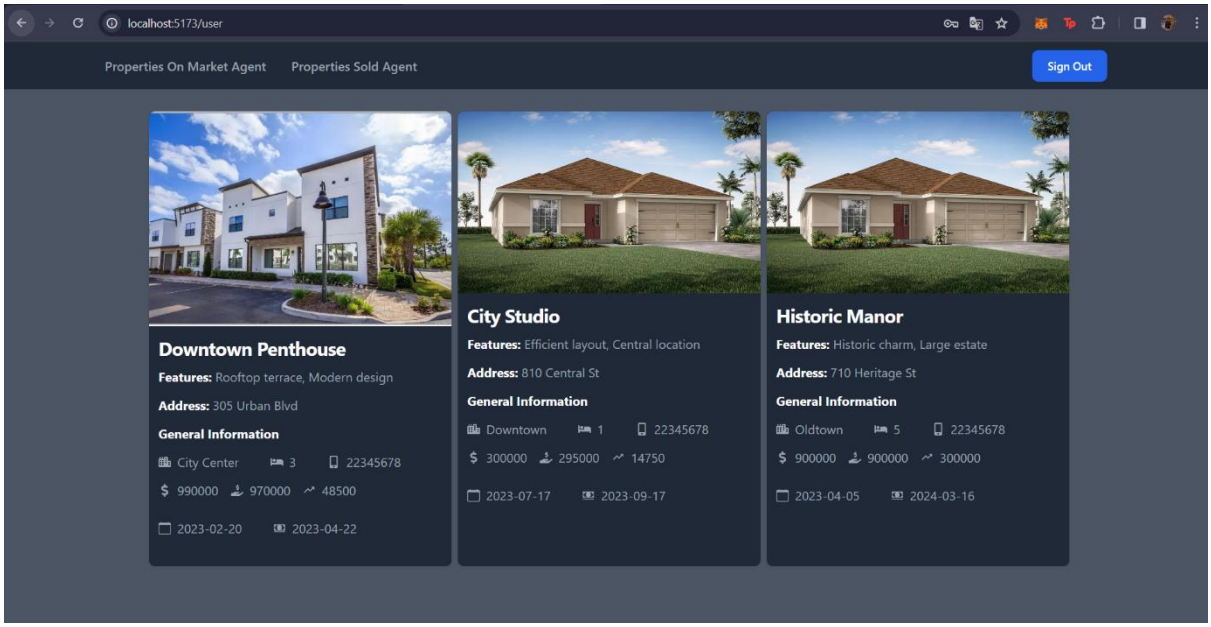
Pantalla de las Propiedades en el mercado:

Click en el botón “Properties on Market Agent”



Pantalla de Propiedades vendidas:

Click en el botón “Properties Sold Agent”

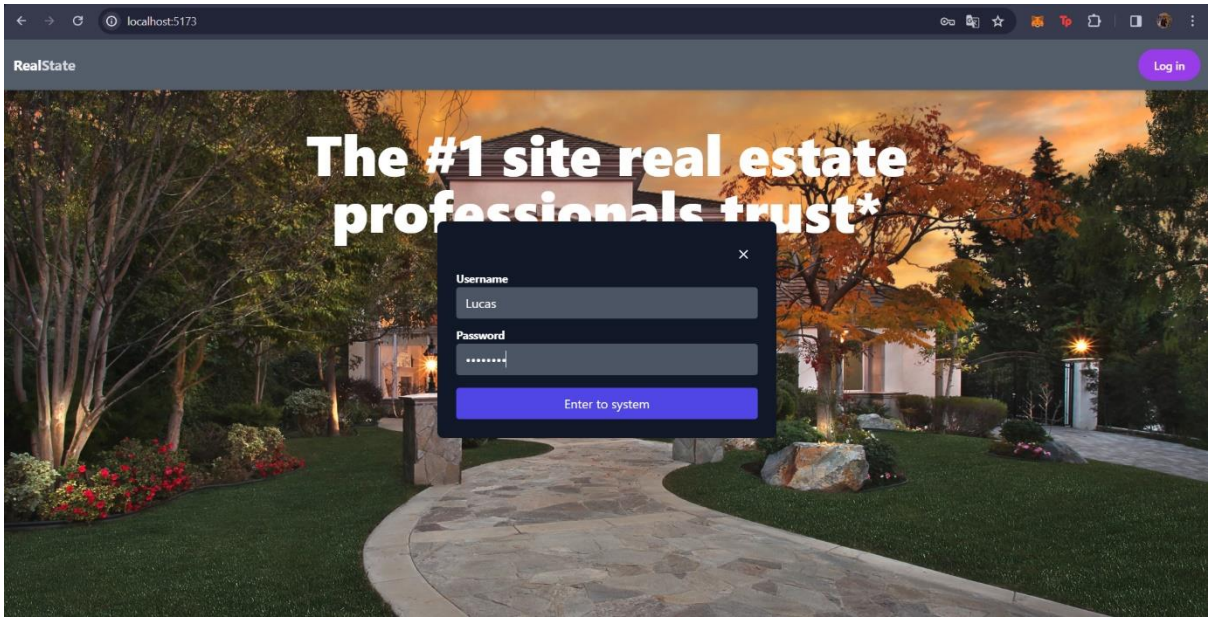


4.1.3. Rol de Comprador

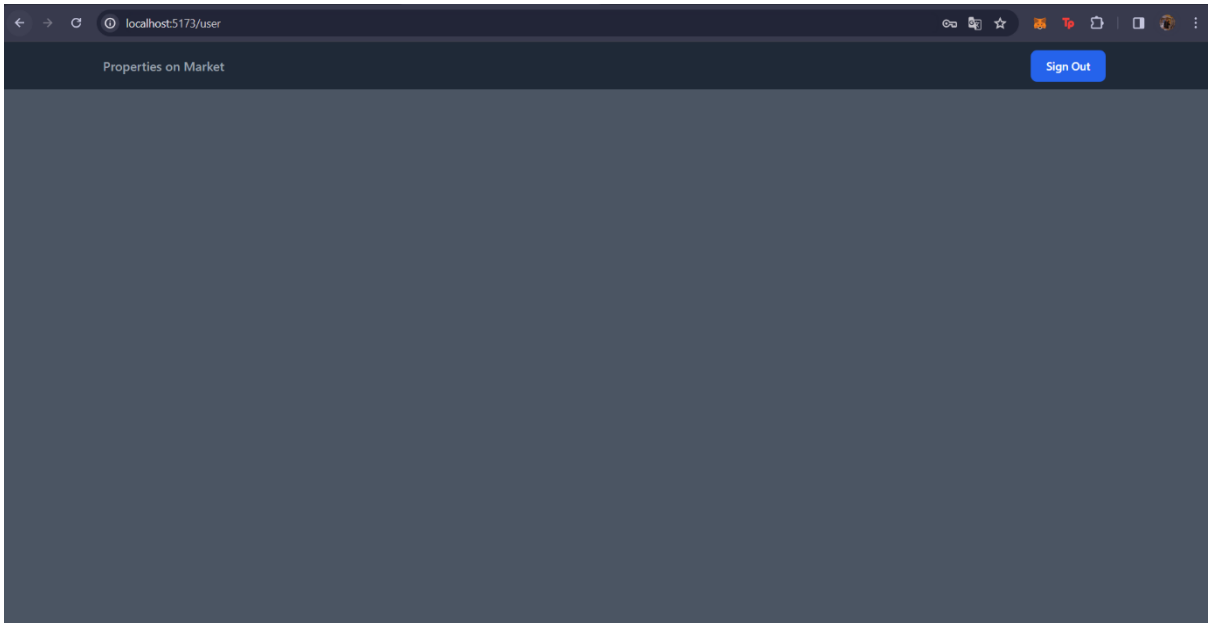
Log In Comprador:

Pasos:

1. Click en el botón “Log In”
2. Ingresar su nombre usuario, en este caso el de los compradores (Lucas)
3. Ingresar la contraseña “12345678”
4. Click en el botón “Enter to System”

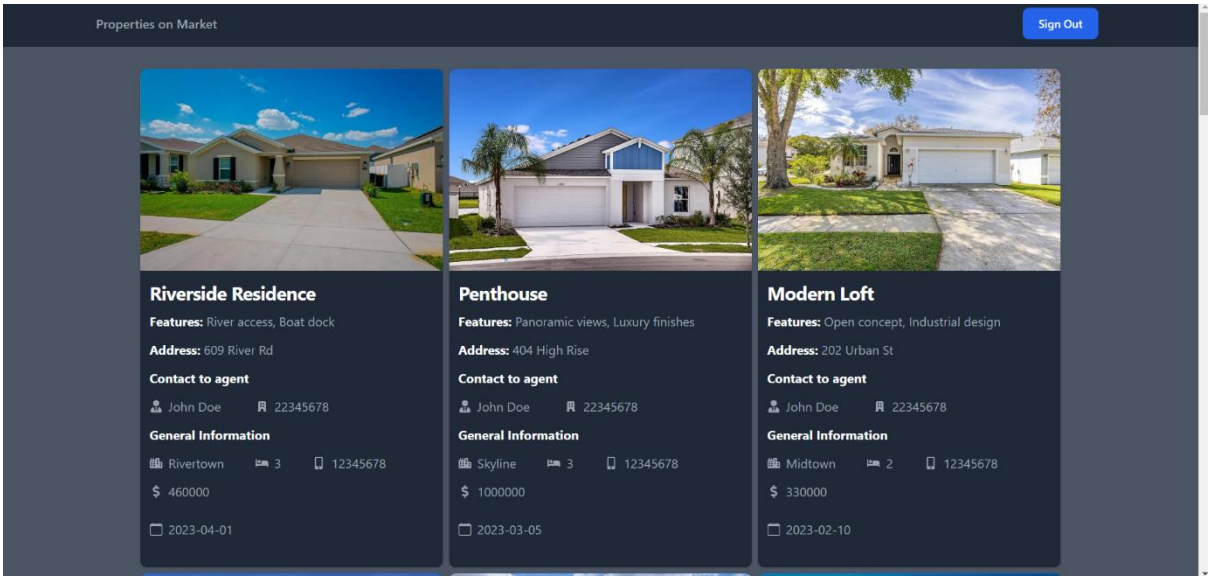


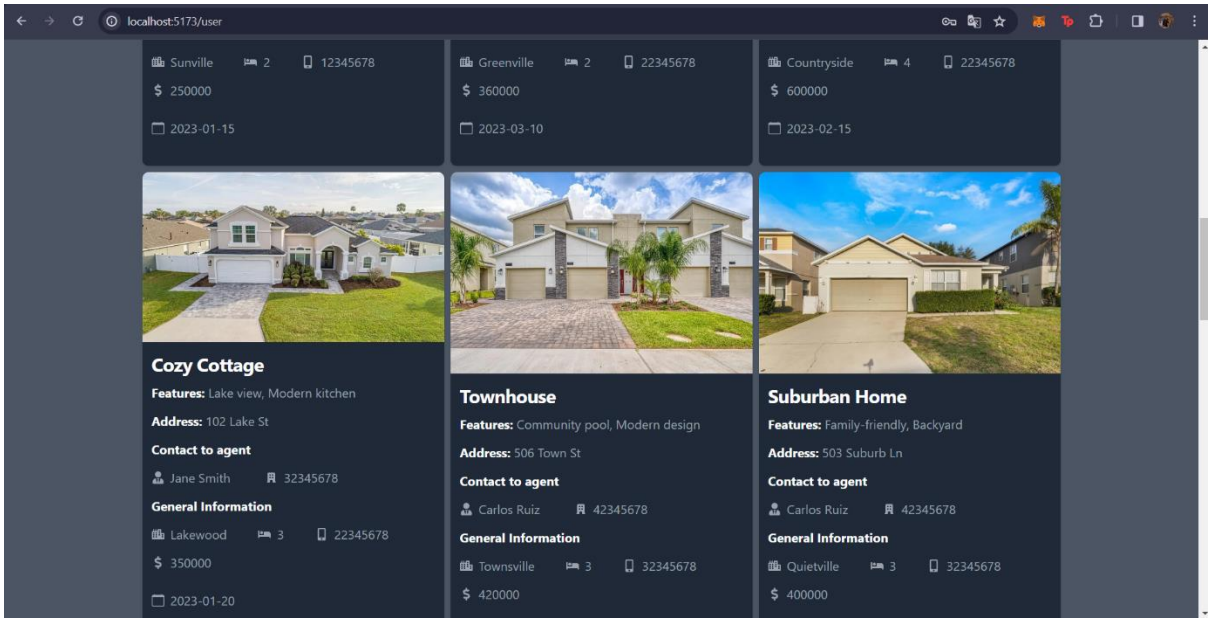
Pantalla Home:



Pantalla Propiedades en el mercado:

Click en el botón “Properties on Market”



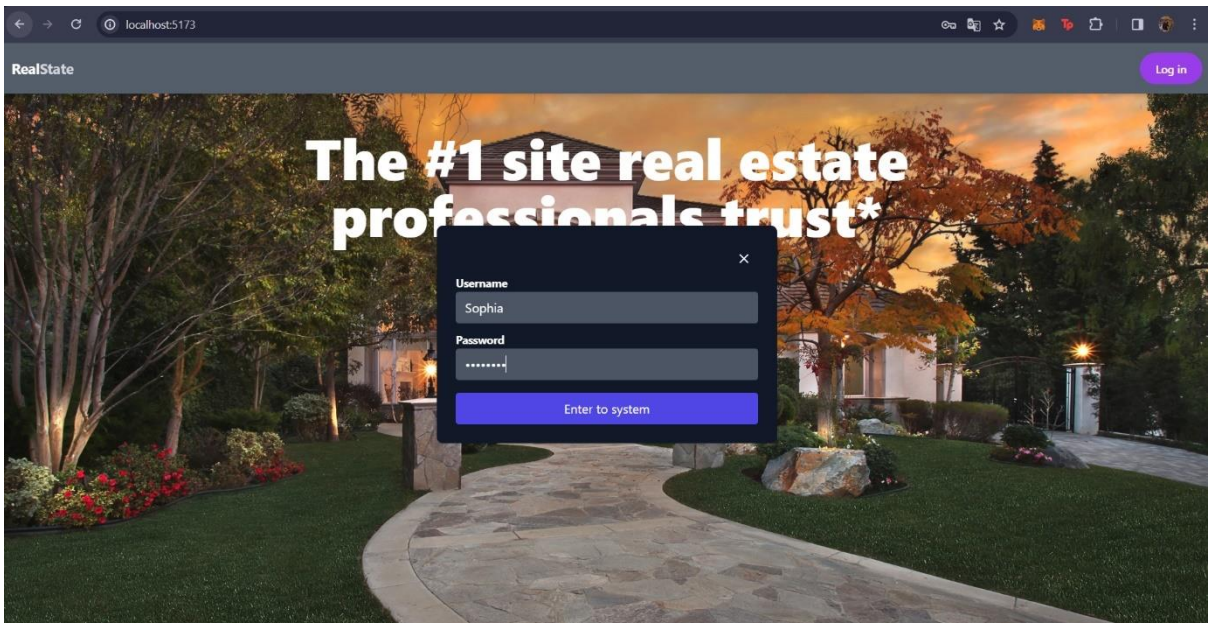


4.1.4. Rol de Vendedor

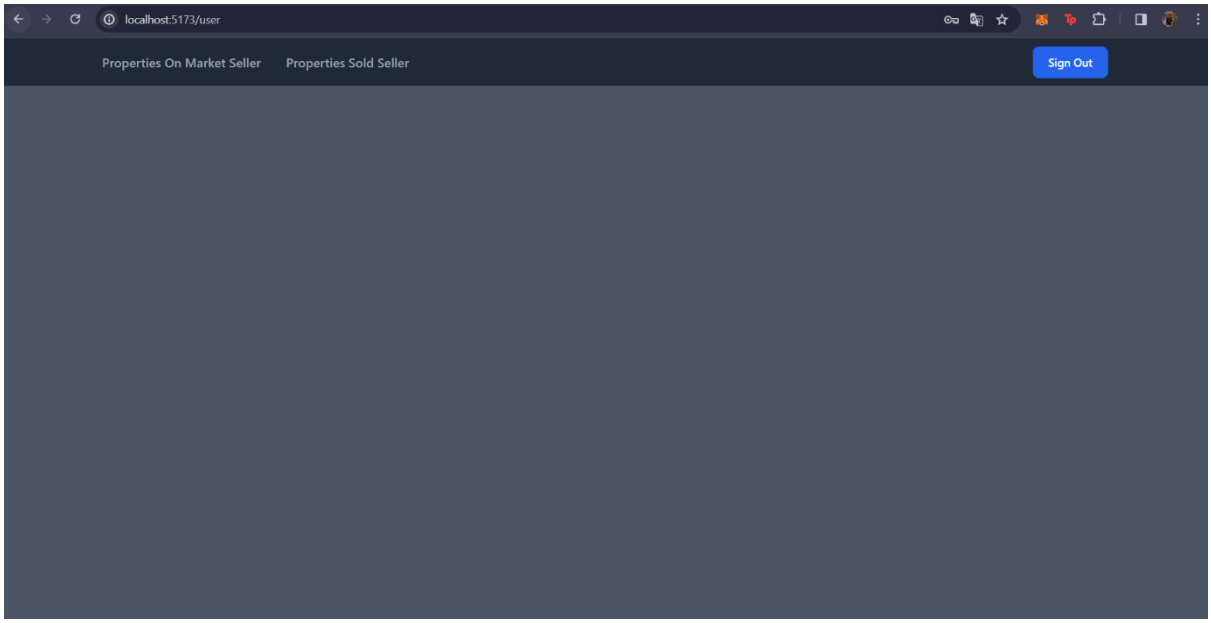
Log In Vendedor:

Pasos:

1. Click en el botón “Log In”
2. Ingresar su nombre usuario, en este caso el de los vendedores (Sophia)
3. Ingresar la contraseña “12345678”
4. Click en el botón “Enter to System”

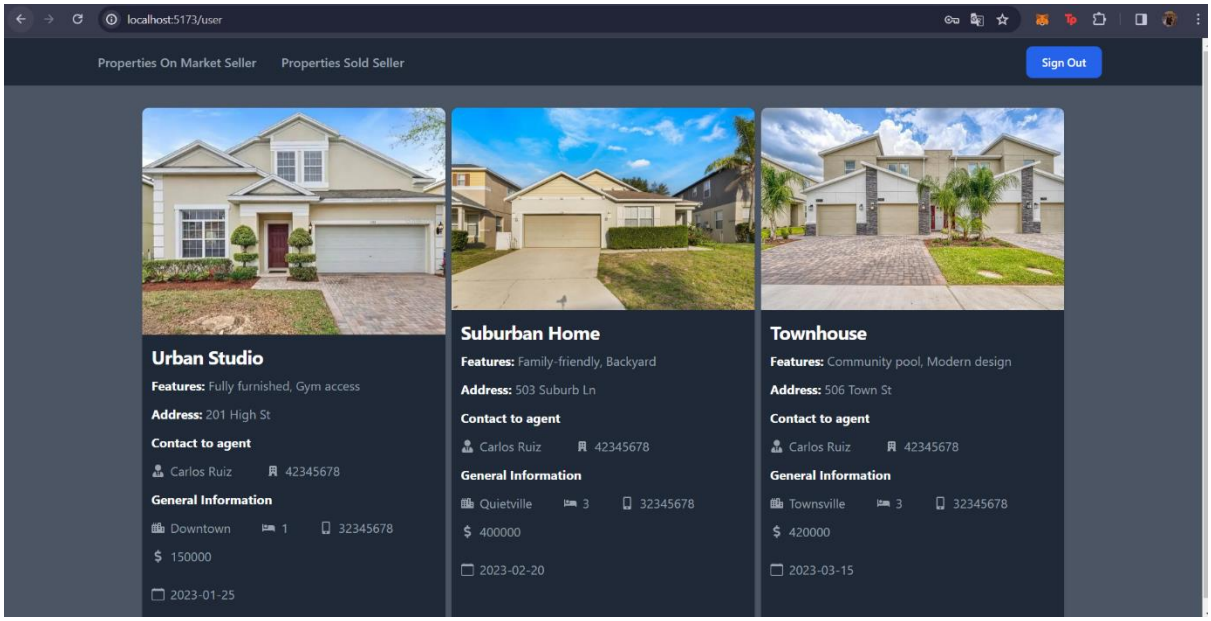


Pantalla Home:



Pantalla de las Propiedades en el mercado:

Click en el botón “Properties on Market Seller”



Pantalla de Propiedades vendidas:

Click en el botón “Properties Sold Seller”