

IES José Mor de Fuentes

Proyecto Final DAW

Curso 2024-2025

Datos del proyecto

Código: IFC_303_numero

Título: Aplicación Gestor de alquileres SECUO

Alumno: Oumar Traore

Profesor Tutor: Modesto Sierra Callau

Convocatoria: Junio



1.- RESUMEN	3
Español	3
English	3
2.- PALABRAS CLAVE	3
Español	3
English	3
3.- INTRODUCCIÓN	3
4.- ANÁLISIS	3
Requisitos funcionales de la aplicación	3
Requisitos no funcionales	4
Requisitos de software	4
Requisitos de hardware	4
Marco legal. Normativa	4
Tecnologías y frameworks	4
5.- PLANIFICACIÓN	4
6.- ESTIMACIÓN DE RECURSOS	4
Humanos	4
Materiales	4
7.- DISEÑO	4
Modelo de datos	5
Diseño funcional	5
Arquitectura de la aplicación	5
Organización del proyecto	5
Plan de pruebas	5
8.- DESARROLLO	5
9.- PLAN DE PRUEBAS	5
10.- MANUAL DE DESPLIEGUE	6
11.- MANUAL DE USUARIO	6
12.- RECURSOS EMPLEADOS	6
13.- ASPECTOS LABORALES Y EMPRESARIALES	7
14.- CONCLUSIONES	8

1.- RESUMEN

Español

SECUO será una app para gestionar reportes de mantenimiento en edificios, permitiendo a inquilinos reportar problemas y a propietarios gestionar las soluciones de manera eficiente. Además, incluirá un sistema de mensajería interna y un historial de reportes y servicios.

Posible mejoras (resaltados en violeta):

Marketplace de viviendas, garajes y muebles en venta o alquiler, que genere ingresos por intermediación en servicios y comisiones de ventas además de las suscripciones premium.

English

SECUO will be an app for managing maintenance reports in buildings, allowing tenants to report issues and landlords to efficiently manage solutions. It will also include an internal messaging system and a history of reports and services.

Possible improvements (highlighted in purple):

A marketplace for homes, garages, and furniture for sale or rent, generating revenue through service intermediation and sales

3.- INTRODUCCIÓN

El proyecto SECUEO surge de la necesidad de una gestión más eficiente de los reportes de mantenimiento en edificios y propiedades. Muchos propietarios tienen dificultades para llevar un control organizado de los problemas reportados por sus inquilinos, lo que retrasa las soluciones y afecta la calidad de vida en los inmuebles. Al mismo tiempo, los inquilinos necesitan una plataforma centralizada donde puedan notificar incidencias de manera sencilla y rápida.

SECUEO está dirigido tanto a propietarios de edificios como a inquilinos. Los primeros podrán recibir reportes de mantenimiento y tendrán un canal eficaz para comunicar cualquier problema en sus viviendas.

Posible mejoras (resaltados en violeta):

También contará con un marketplace de bienes inmuebles y mobiliario, donde los usuarios podrán buscar viviendas, garajes y muebles en venta o alquiler. Este enfoque integral no solo facilita la gestión de propiedades, sino que también abre oportunidades comerciales dentro de la plataforma.

SECUEO generará ingresos mediante la intermediación en la contratación de servicios de mantenimiento, la implementación de planes de suscripción premium para funciones avanzadas, y la comisión por la compraventa de muebles e inmuebles. Su objetivo es convertirse en una herramienta esencial para la administración de propiedades, optimizando la comunicación entre inquilinos y propietarios y proporcionando soluciones rápidas y efectivas a los problemas de mantenimiento.

4.- ANÁLISIS

Facilita la comunicación entre inquilinos y propietarios. Además, ofrece un marketplace inmobiliario donde los usuarios pueden comprar, vender o alquilar viviendas, garajes y muebles.

Funcionalidades principales

1. Gestión de reportes de mantenimiento

- Los inquilinos pueden registrar incidencias (goteras, fallos eléctricos, etc.).
- Los propietarios pueden visualizar los reportes en tiempo real y gestionar su resolución.
- Notificaciones para mantener informadas a ambas partes sobre el estado de las incidencias.
- Historial de reportes para seguimiento de incidencias anteriores.
- Los usuarios pueden switchear entre modo inquilino o modo propietario para ver las diferentes vistas y opciones que ofrece la aplicación.

2. Sistema de mensajería:

- Mensajería interna entre inquilinos y propietarios para facilitar la comunicación.
- Notificaciones en tiempo real.

3. Monetización

- Modelos de suscripción premium con funcionalidades avanzadas.
- Porcentaje sobre las ventas y alquileres gestionados dentro de la plataforma.

4. Marketplace inmobiliario

- Publicación y búsqueda de viviendas, garajes y muebles en venta o alquiler.
- Filtros avanzados para encontrar propiedades según ubicación, precio y características.

El objetivo de esta aplicación es la de optimizar la administración de inmuebles, centralizando reportes, contrataciones y compraventas en una sola aplicación eficiente y fácil de usar.

Requisitos funcionales de la aplicación

Gestión de reportes de mantenimiento:

- Gestión de reportes de mantenimiento.
- Notificaciones automáticas a propietarios e inquilinos.
- Historial de reportes y servicios.
- Sistema de mensajería interna.
- Monetización a través de suscripciones premium.

ROLES DE USUARIOS

Usuario -modo Inquilino

- Registrar reportes de mantenimiento y dar seguimiento a su estado.
- Recibir notificaciones sobre la resolución de incidencias.
- Acceder al historial de reportes y **servicios contratados**.
- Enviar mensajes a propietarios a través del sistema de mensajería interna.

Usuario- modo Propietario

- Recibir y gestionar reportes de mantenimiento de sus propiedades.
- **Contratar servicios de reparación y mantenimiento dentro de la plataforma.**
- Acceder al historial de reportes y servicios contratados.
- Enviar y recibir mensajes con inquilinos a través del sistema de mensajería.

Administradores

- **Gestionar usuarios y supervisar la actividad en la plataforma.**
- **Controlar el sistema de pagos y comisiones.**
- **Velar por el cumplimiento de normativas y términos de uso.**

Requisitos no funcionales

Requisitos de software

- Backend desarrollado en **Spring Boot**.
- Frontend en **Angular, CSS, HTML, Bootstrap**.
- Base de datos en **MySQL, MongoDB**.
- Servidor web con **Apache** o **Nginx**.
- Sistema operativo en servidor basado en **Linux (Ubuntu o CentOS)**.

Requisitos de hardware

- Estimación de recursos hardware necesarios. Se puede hacer una estimación de recursos en función de usuarios si se considera necesario
- Servidor con al menos **4 vCPUs y 8GB de RAM**.
- Almacenamiento de al menos **100GB SSD**.
- Capacidad de escalar en función del crecimiento de usuarios.

Marco legal. Normativa

- Cumplimiento con GDPR para la protección de datos de los usuarios.
- Política de privacidad y términos de uso claros para los usuarios.
- **Normativas de comercio electrónico en caso de integrar pagos.**

Tecnologías y frameworks

Frameworks y principales tecnologías a utilizar en el desarrollo y justificación de su elección.

- **Backend:** Spring Boot (seguridad, escalabilidad y comunidad activa).
- **Frontend:** Bootstrap, Angular (modularidad y compatibilidad con REST APIs).
- **Base de datos:** MySQL y MongoDB.
- **Servidor:** Apache/Nginx (rendimiento y flexibilidad).
- **Notificaciones:** Firebase o WebSockets.
- **Mensajería interna:** WebSockets o Firebase Cloud Messaging.

5.- PLANIFICACIÓN

Esta sección detallo el cronograma del proyecto SECUO, estableciendo las fases clave, fechas y horas previstas para su desarrollo, desde el análisis hasta el despliegue, con un total de 99 horas distribuidas en aproximadamente.

Fase	Fecha inicio	Fecha fin	Horas previstas
<i>Análisis</i>	10/03/2025	15/03/2025	12
<i>Diseño</i>	20/03/2025	31/03/2025	12
<i>Codificación</i>	31/03/2025	20/04/2025	50
<i>Backend</i>	31/03/2025	12/04/2025	35
<i>Frontend</i>	12/04/2025	20/04/2025	15
<i>Pruebas</i>	21/04/2025	30/04/2025	15
<i>Despliegue</i>	1/05/2025	05/05/2025	10

[Enlace Diagrama de Gantt](#)

Hitos principales:

- Finalización del análisis y diseño (31/03/2025).
- Desarrollo de backend y frontend (20/04/2025).
- Integración de funcionalidades principales (20/04/2025).
- Pruebas de la aplicación (30/04/2025).
- Lanzamiento y despliegue (05/05/2025).

6.- ESTIMACIÓN DE RECURSOS

Humanos

La siguiente tabla desglosa las horas planificadas por perfil profesional según la sección de planificación, asignando costos por hora basados en una el análisis en tarifas promedio en España y calculando el total estimado para los recursos humanos del proyecto:

Perfil	Horas	Costo por hora	Total
Desarrollador Backend	35	35	1.225€
Desarrollador Frontend	15	35	525€
Diseñador UI/UX	15	30	450€
Tester	15	30	450€
DevOps	10	45	450€
Total estimado	90	-	3.100€

Desarrollador Backend: Desarrolla APIs, bases de datos y lógica de reportes/mensajería.

Desarrollador Frontend: Crea interfaz y vistas con Angular para usuarios.

Diseñador UI/UX: Diseña UX/UI y prototipos visuales.

Tester: Prueba funcionalidad, estabilidad y corrige errores.

DevOps: Configura servidor, despliegue y escalabilidad.

Materiales

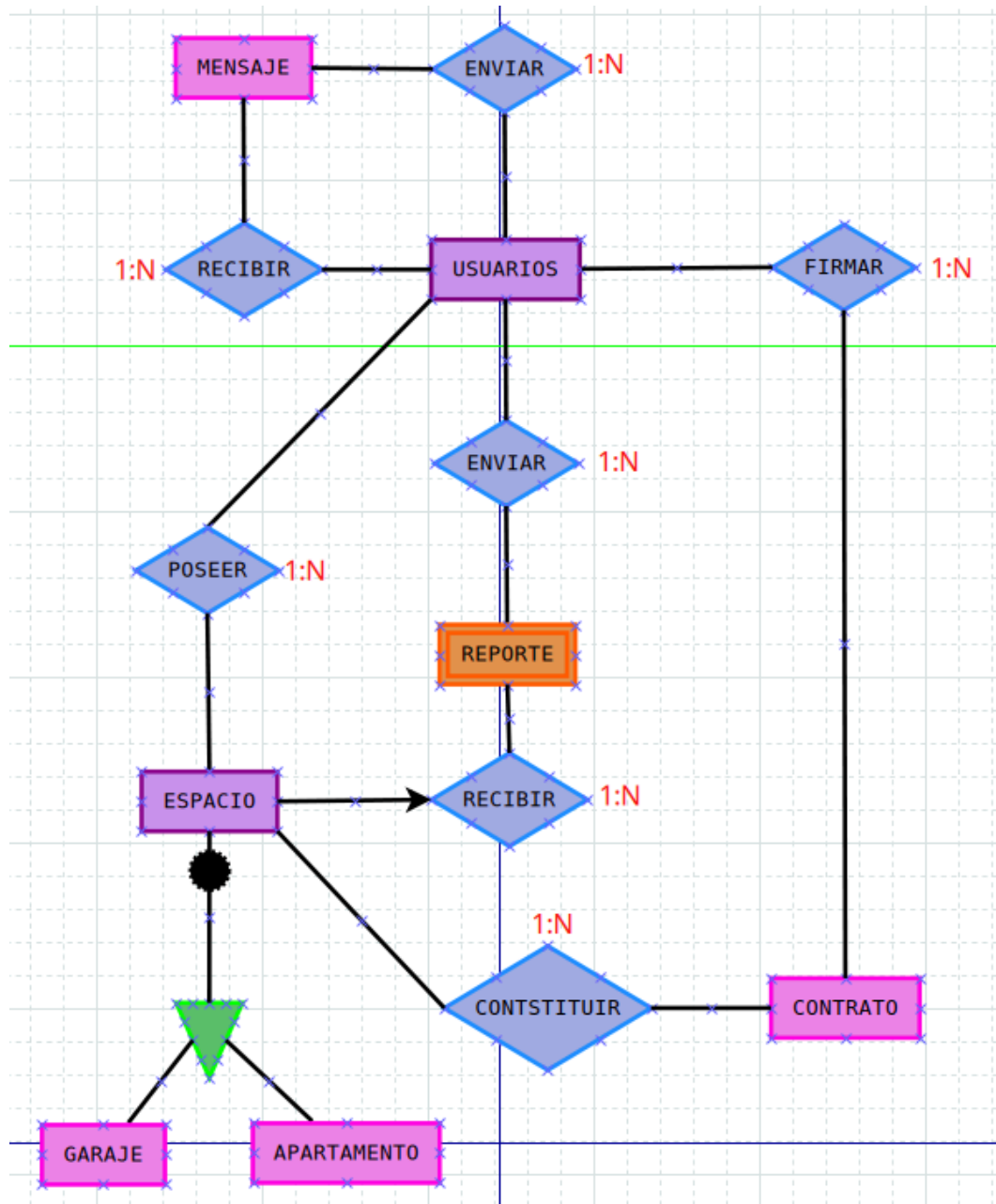
Esta tabla presenta los recursos materiales necesarios para el desarrollo y operación de SECUO durante los primeros 3 meses, estimando costos de hardware, software, material de oficina y otros elementos esenciales según las necesidades de mi proyecto:

Recurso	Descripción	Costo
Servidor cloud	50 €/mes × 3	150€
Equipo de desarrollo	5 equipos × 800 €	4.000€
Dispositivos de prueba	2-3 móviles/tablets	500€
Herramientas desarrollo	IntelliJ, 20 €/mes × 3	60€
Herramientas diseño	Figma, 12 €/mes × 3	36€
Gestión de proyectos	Jira, 10 €/mes × 3	30€
Suministros oficina	Papel, bolígrafos	50€
Impresora	30 €/mes × 3	90€
Espacio de trabajo	Coworking, 100 €/mes × 3	300€
Dominio y SSL	Dominio + certificado	65€
Mantenimiento	20 €/mes × 3	60€
Total estimado	(Con equipos nuevos)	5.341€

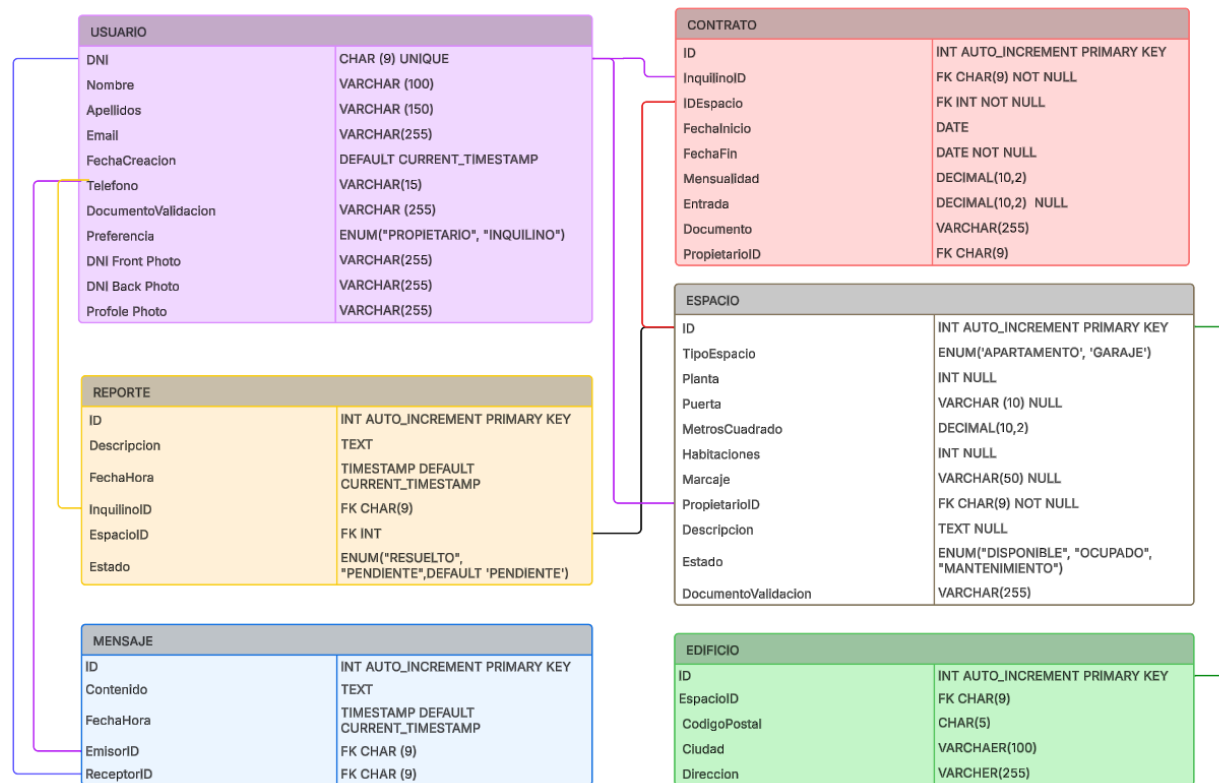
7.- DISEÑO

Modelo de datos

MODELO ENTIDAD RELACIÓN



MODELO RELACIONAL



Enlace para visualizar mejor la tabla y los atributos: [LUCID](#)


Relaciones:


- Un **Usuario** (Propietario) puede tener muchos **Espacios** (PropietarioID).
- Un **Espacio** pertenece a un **Edificio** (EspacioID).
- Un **Contrato** vincula a un **Inquilino** (Usuario) con un **Espacio**.
- Un **Reporte** está asociado a un **Inquilino** y un **Espacio**.
- Un **Mensaje** se envía entre un **Emisor** y un **Receptor** (ambos Usuarios).

Diseño funcional

Diseño de interfaz de la aplicación: **Login View**

SECUO Propiery's Manager





Welcome to SECUO
an innovative, scalable, and efficient platform for seamless rental management. SECUO empowers landlords and tenants with smart tools to manage properties, streamline communication, and handle maintenance effortlessly.

With a user-friendly interface and integrated messaging system, SECUO transforms property management into a simple, transparent, and effective experience. 🚀

LOGIN

User DNI


Password


Enter as: OWNER ☐

[I havent account, create now](#)

Register View

SECUO Propiery's Manager





Welcome to SECUO
an innovative, scalable, and efficient platform for seamless rental management. SECUO empowers landlords and tenants with smart tools to manage properties, streamline communication, and handle maintenance effortlessly.

With a user-friendly interface and integrated messaging system, SECUO transforms property management into a simple, transparent, and effective experience. 🚀

REGISTER

DNI

Name

Surname

Email

Phone

Front Photo DNI

Back Photo DNI

Password

Repeat psswd

[I alredy have account, login now](#)

Inquilino - Welcome / Dashboard + LIGHTMODE View



Propietario - Welcome / Dashboard + DARKMODE View



Propietario - Reports View

P Welcome Mario

Propiedad 2 - Piso Calle Modesto 72 N°3

-  CUCARACHAS EN LA COCINA!!! 4
-  NO FUNCIONA LA DUCHA, NO HAY AGUA 2
-  Buenas noches, alguno más te ha avisado sobre el agua o solo es amí?... 1
-  Voy a mudarme por temas de trabajo, cuándo te viene bien para arreglar los papeles? 1

Mario Sosal Lapispazuli

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled.

Logout

Propietario - Messages View

P Welcome Mario

CUCARACHAS EN LA COCINA!!!

En 20 años nunca han habido bichos sserá que usted no tiene buena higiene, voy a llamar a los antiplaga

PERDONA PERO ESTA FALTA DE RESPETO NO LA VOY A TOLERAR!

DESDE LA PRMERA SEMANA HABÍA VISTO UNA RATA EN LA ENTRADA, SU PROPIEDAD SIEMRE FUE UNA COLONIA DE PLAGAS

Señora deje de escribir en mayuscula que eso no le va a dar la razón

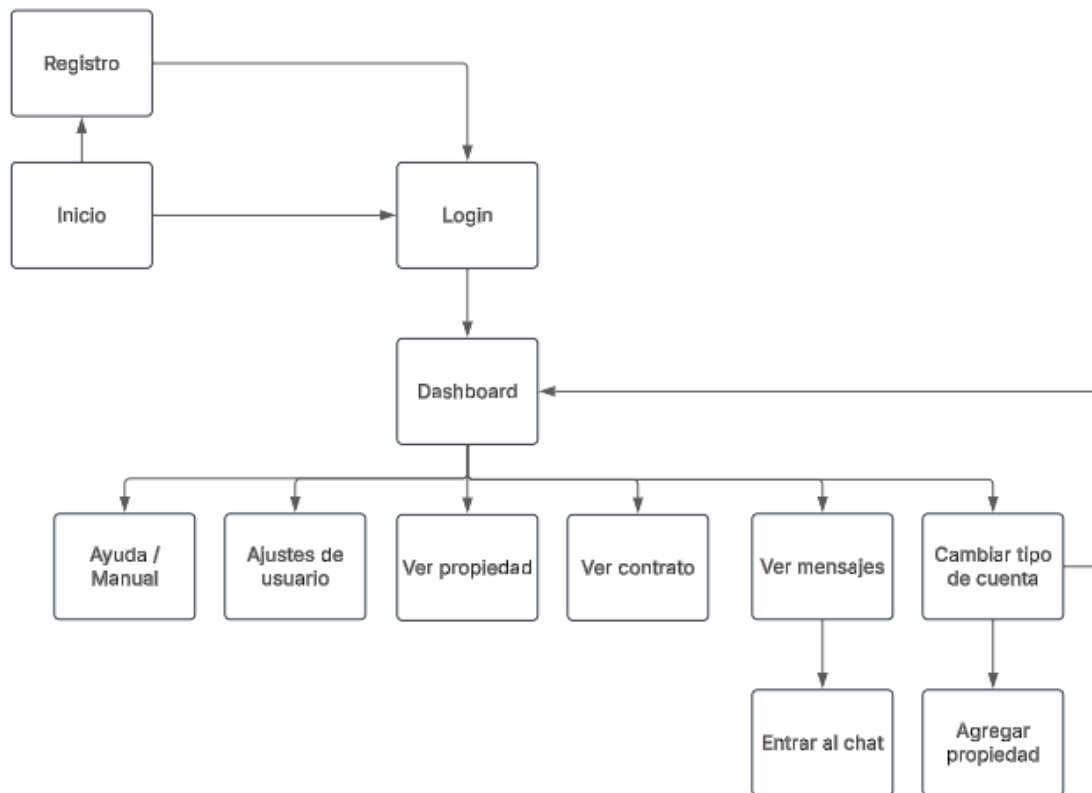
Si no le gusta el piso, puede recoger sus cosas y largarse, gracias |

Mario Sosal Lapispazuli

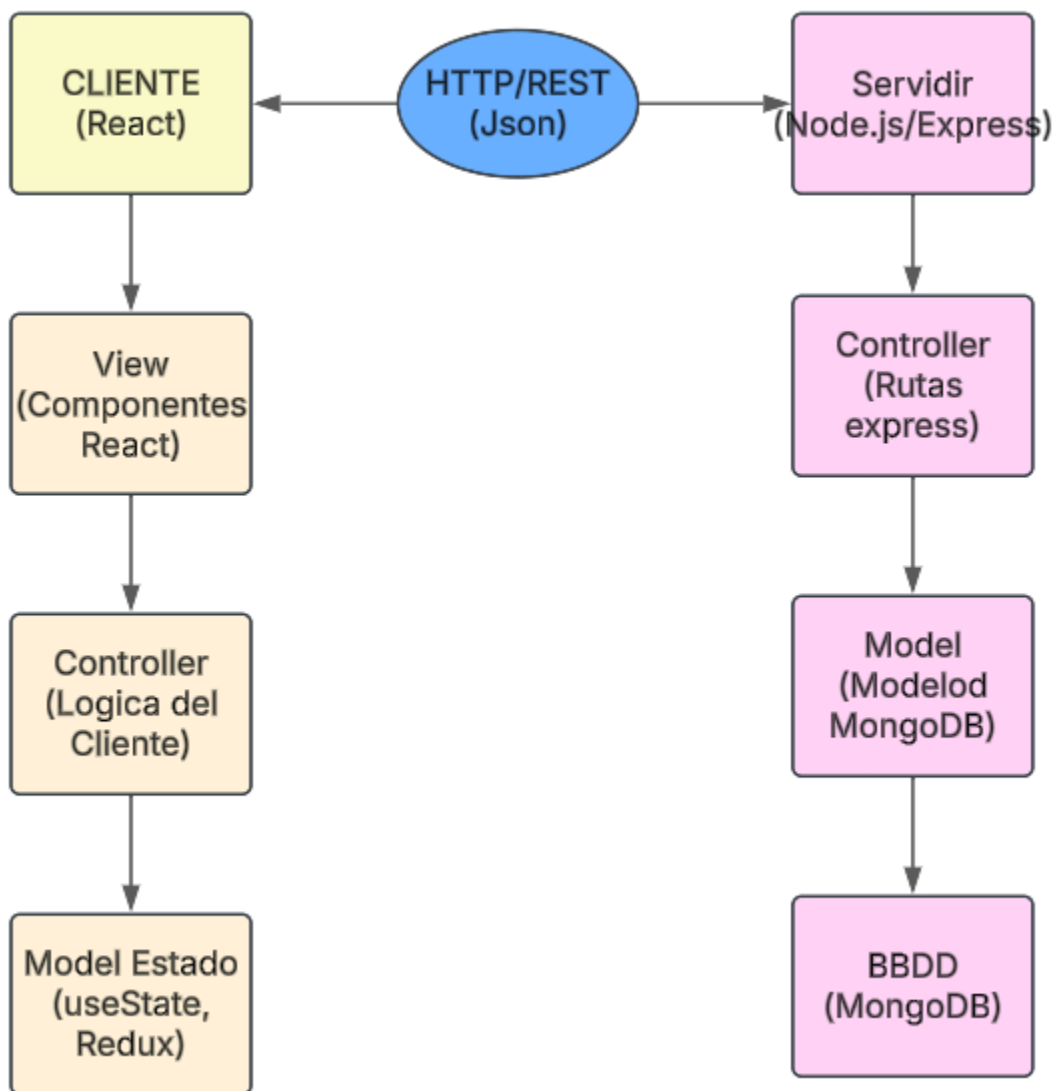
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled.

Logout

Árbol de navegación:



Arquitectura de la aplicación



Cliente (React): Maneja la interfaz de usuario y la lógica del cliente.

Servidor (Node.js/Express.js): Maneja las solicitudes HTTP, la lógica de negocio, y el acceso a la base de datos.

Interacciones: Comunicación RESTful (HTTP/JSON).

División cliente y servidor. Componentes principales en cada parte e interacción entre ellos. Se debería respetar en la medida de lo posible alguna arquitectura conocida.

3. Componentes Principales e Interacciones

Servidor (Node.js/Express.js)

En el servidor, aplicaremos el patrón MVC de manera más estricta:

- **Model (Modelo):**
 - Representa los datos y la lógica de acceso a la base de datos.
 - Ejemplo: Modelos de Mongoose para Usuario, Espacio, etc.
- **View (Vista):**
 - No aplica directamente, ya que es una API REST (el cliente React maneja la vista).
 - Consideramos las respuestas JSON como una "vista" en este contexto.
- **Controller (Controlador):**
 - Maneja las solicitudes HTTP y coordina las interacciones entre el modelo y la respuesta.
 - Ejemplo: Controladores para manejar las rutas /api/usuarios.

Cliente (React)

En REACT no se aplica MVC de manera estricta como en el servidor, pero lo estructura de forma que se asemeja al patrón para mantener la organización.

- **View (Vista):**
 - Componentes de React que renderizan la interfaz de usuario.
 - Ejemplo: UserList.js (muestra una lista de usuarios), UserForm.js (formulario para crear un usuario).
- **Controller (Lógica del Cliente):**
 - Lógica para manejar eventos de usuario y comunicarse con el servidor.
 - Ejemplo: Funciones que hacen peticiones HTTP al servidor usando axios o fetch.
- **Model (Estado):**
 - Estado de la aplicación, manejado con useState o una librería como Redux.
 - Ejemplo: Un estado que almacena la lista de usuarios obtenida del servidor.

Organización del proyecto

Estructura de carpetas y ficheros

```
frontend/
├── src/
│   ├── components/ # Reusable components (buttons, inputs, etc.)
│   ├── pages/ # Main pages (Login, Chat, Profile)
│   ├── hooks/ # Custom React hooks
│   ├── context/ # Context API for global state management
│   ├── services/ # API calls (Axios)
│   ├── styles/
│   │   ├── components/ # Components CSS styles
│   │   ├── pages/ # Pages CSS styles
│   │   └── others/ # General utilities or context styles
│   ├── socket/ # WebSocket connections
│   ├── App.jsx # Main app component
│   └── main.jsx # React entry point
├── public/ # Static files
├── package.json # Dependencies config
└── vite.config.js # Vite config
```

Link Version Backend Visual [LUCID](#)

Link Version Frontend Visual [LUCID](#)

Plan de pruebas

Listado de pruebas a realizar para garantizar el cumplimiento de los requisitos funcionales

1.1. Usuarios

- **Listar Usuarios (GET /api/usuarios):** El servidor devuelve una lista de usuarios, y el cliente la muestra en una tabla.
- **Crear un Usuario (POST /api/usuarios):** El servidor crea un usuario, y el cliente envía los datos y muestra un mensaje de éxito o error.
- **Manejo de Errores:** El servidor y el cliente manejan errores como DNI duplicado o datos inválidos.

1.2. Contratos de Alquiler

Listar Contratos de Alquiler (GET /api/contratos): El servidor devuelve una lista de contratos, y el cliente la muestra en una tabla.

Crear un Contrato de Alquiler (POST /api/contratos): El servidor crea un contrato, y el cliente envía los datos y muestra un mensaje de éxito o error.

Actualizar un Contrato (PUT /api/contratos/:id): El servidor actualiza un contrato (por ejemplo, cambiar el estado a "finalizado").

Eliminar un Contrato (DELETE /api/contratos/:id): El servidor elimina un contrato.

Manejo de Errores: Por ejemplo, no se puede crear un contrato si el espacio ya está ocupado o si el inquilino/propietario no existe.

1.3. Contratos Asignados

- **Listar Contratos Asignados a un Usuario (GET /api/usuarios/:id/contratos):** El servidor devuelve los contratos asociados a un usuario (como inquilino o propietario), y el cliente los muestra.
- **Manejo de Errores:** Por ejemplo, devolver un error si el usuario no existe o no tiene contratos asignados.

1.4. Reportes

- **Crear un Reporte (POST /api/reportes):** El inquilino crea un reporte sobre un problema, y el servidor lo guarda y lo asocia al contrato correspondiente.
- **Listar Reportes por Contrato (GET /api/contratos/:id/reportes):** El servidor devuelve los reportes asociados a un contrato, y el cliente los muestra (puede ser visto por el inquilino o el propietario).
- **Listar Reportes Recibidos por un Propietario (GET /api/propietarios/:id/reportes):** El servidor devuelve todos los reportes dirigidos a un propietario (basado en los contratos donde es propietario), y el cliente los muestra.
- **Actualizar un Reporte (PUT /api/reportes/:id):** El propietario actualiza el estado del reporte (por ejemplo, de "pendiente" a "resuelto").
- **Manejo de Errores:** Por ejemplo, un inquilino no puede crear un reporte para una vivienda que no le pertenece.

8.- DESARROLLO

Decisiones e incidencias durante el periodo de desarrollo / codificación

Decisiones

- Poner en la misma vista la posibilidad de ver propiedades propias y ajenas donde somo inquilinos
- JWT para la autenticación por ser una solución ligera y fácil de integrar tanto en backend como en frontend.
- Que haya administradores de promedio para gestionar el app y los datos y las validaciones, como cuando se registra un nuevo usuario y pone su dni, esto llegara a un admin que revisara el dni y las imagenes que suba para verificarlo, lo mismo con los contratos y las propiedades, adjuntar un documento para validar
- Guardar los datos en local para hacerlo más rápidamente, más tarde si me da tiempo uso el cloud de AWS

Incidencias

- Datos que paso de una vista a otra son null o undefined
- Rutas api no encontradas status 400 Bad Request múltiples razones
- Errores de acceso y redireccionamiento, tanto en backend por api como en frontend por rutas
- Variable definido como array de objetos intentando acceder a un array string
- Rutas de destino absolutas en las cargas de los ficheros e imágenes aportadas
- Errores de autenticación porque no tengo token para acceder a x página o usar ruta api
- Errores al usar diferentes componentes de react porque me falta pasarle un prop o porque no es una función
- Errores al equivocarme cuando extraía o trataba de usar datos del params o del body
- Sobreescritura de componentes envueltos y componentes al usar css genérico sin entrar en más detalle las clases e ids de los componentes html
- Imposibilidad de instalar y usar Tailwind en vez de CSS puro

Repositorio donde se aloja código. Versiones generadas

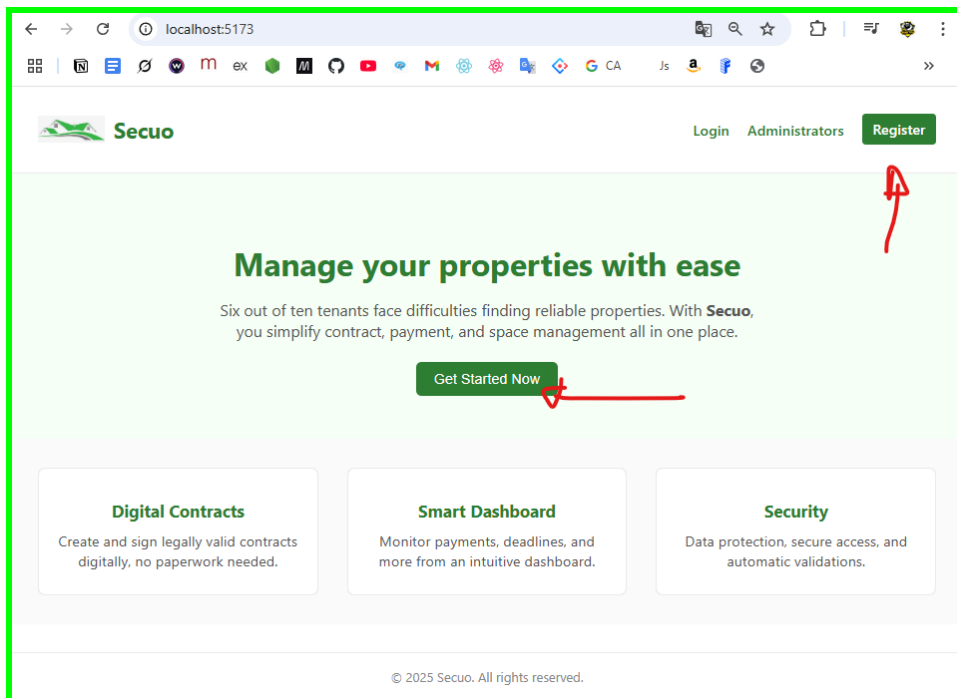
[SecuoGithub](#)

9.- PLAN DE PRUEBAS

Evidencias de ejecución del plan de pruebas, con fecha y resultado

Prueba	Requeto Funcional	Fecha	Resultado
POSTMAN API USERS	Funcionamiento perfecto de todas las rutas api de usuario	##	(OK/NO OK/Parcial).
POSTMAN API SPACES	Funcionamiento perfecto de todas las rutas api de Spaces	##	OK
POSTMAN API CONTRACTS	Funcionamiento perfecto de todas las rutas api de usuario	##	OK
POSTMAN API ADMINS	Funcionamiento perfecto de todas las rutas api de admin	##	OK
POSTMAN API USERS	Funcionamiento perfecto de todas las rutas api de usuario	##	OK

Prueba #001 Crear un nuevo usuario



La contraseña es corta porque hago muchos test, más tarde los pongo a medida

The screenshot shows a 'User Registration' form. The form is titled 'User Registration' and contains the following fields:

- DNI * (123123123)
- First Name * (Modesto)
- Last Name * (Sierra Sierra Callau)
- Email * (modes@sierra.com)
- Phone Number * (677835655)
- Role (Owner)
- Password * (...)

Upload your DNI photo - Front *

Drag and drop FROT photo here, or click to select

File: mern.png

Upload your DNI photo - Back *

Drag and drop BACK photo here, or click to select


File: mernw.png

Upload your profile photo *

Drag and drop PROFILE photo here, or click to select

File: hq720.jpg

Register

 SECUCO Property Manager

Welcome to SECUCO

A scalable and efficient platform for seamless rental management. Empowering landlords and tenants with tools to streamline communication, handle maintenance, and manage properties effortlessly.

Experience simple, transparent, and effective property management.

Login

User DNI

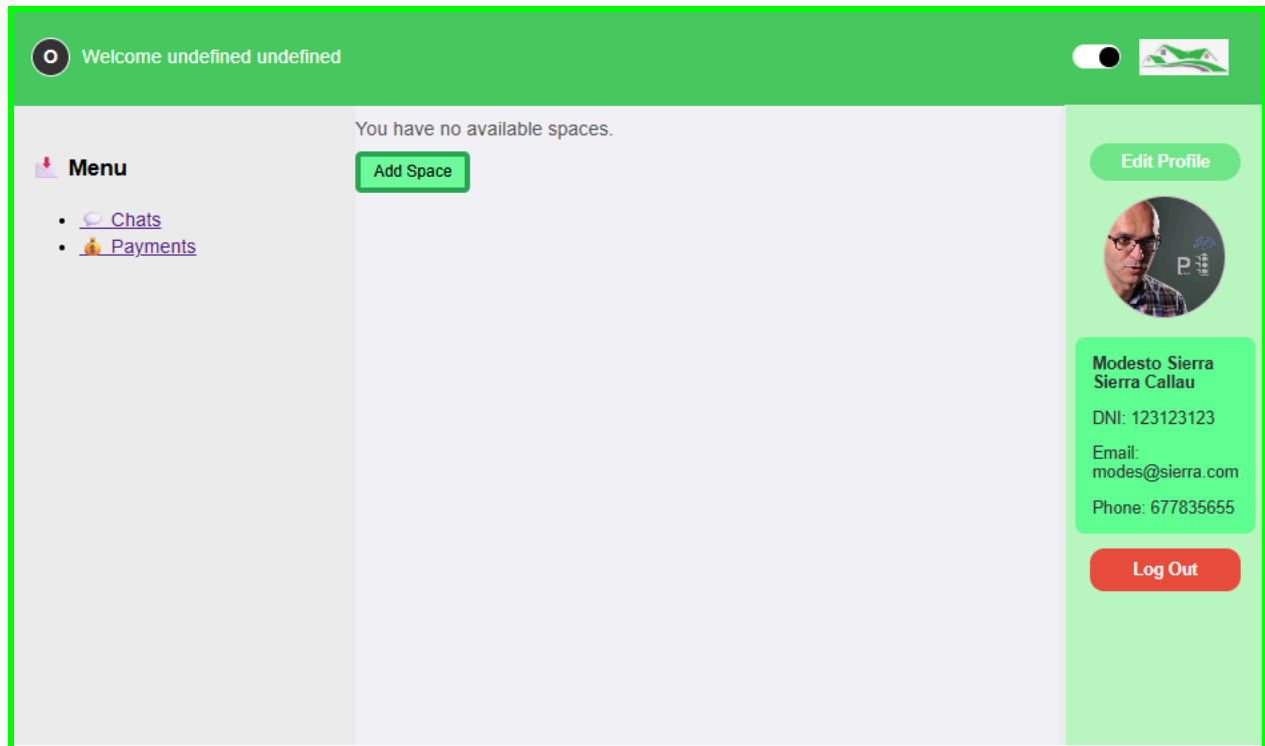
123123123

Password

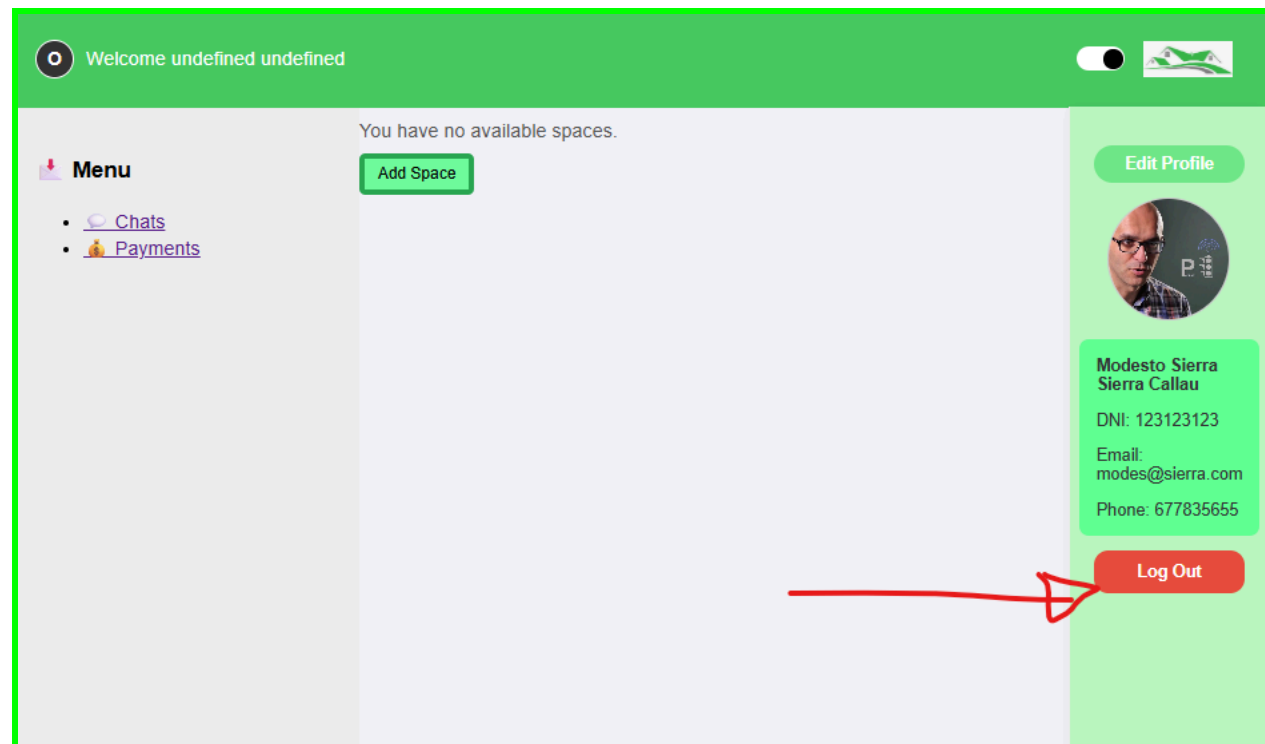
...

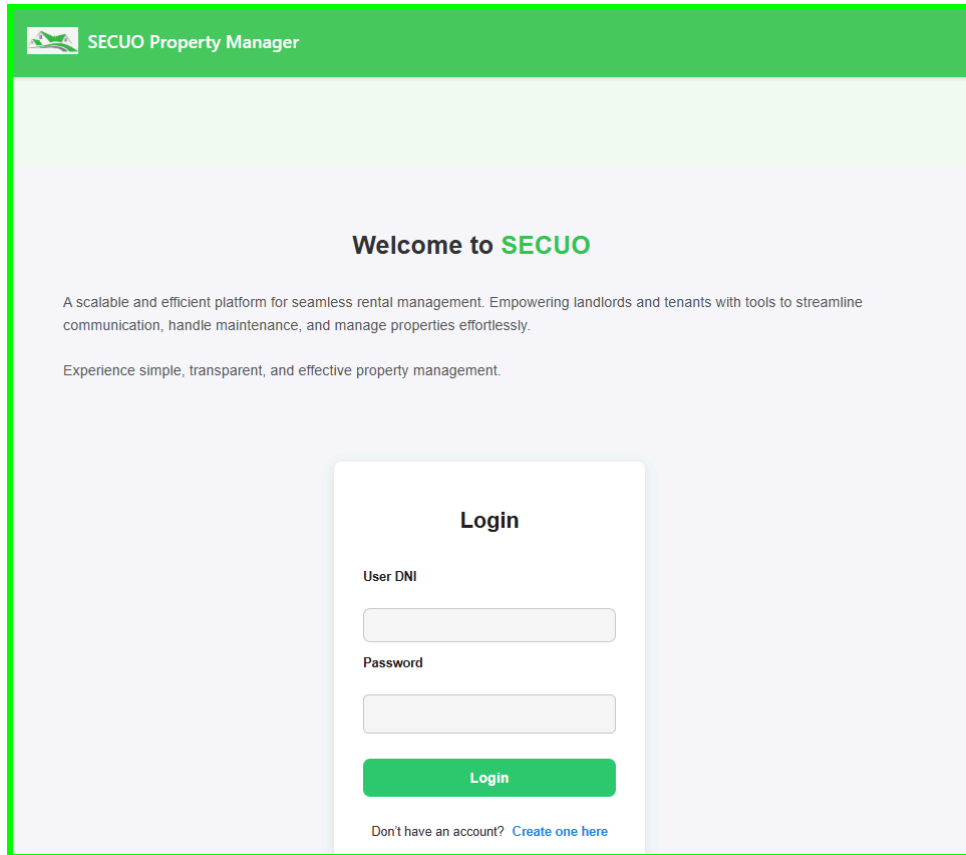
Login

Don't have an account? [Create one here](#)



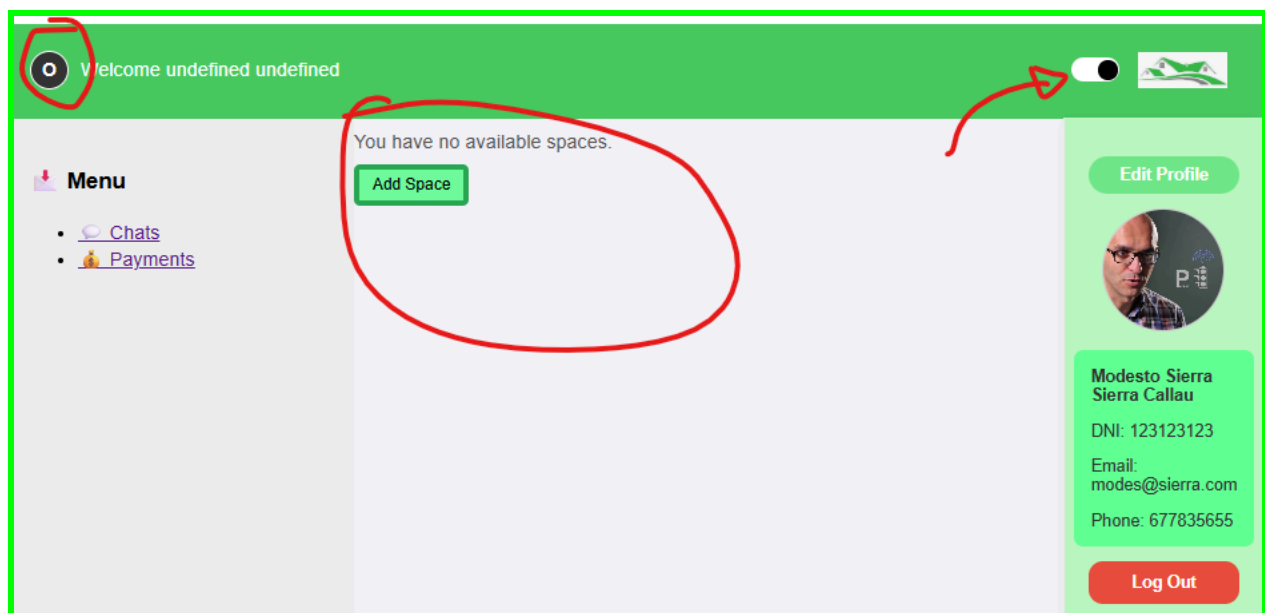
Prueba #002 Cerrar sesión



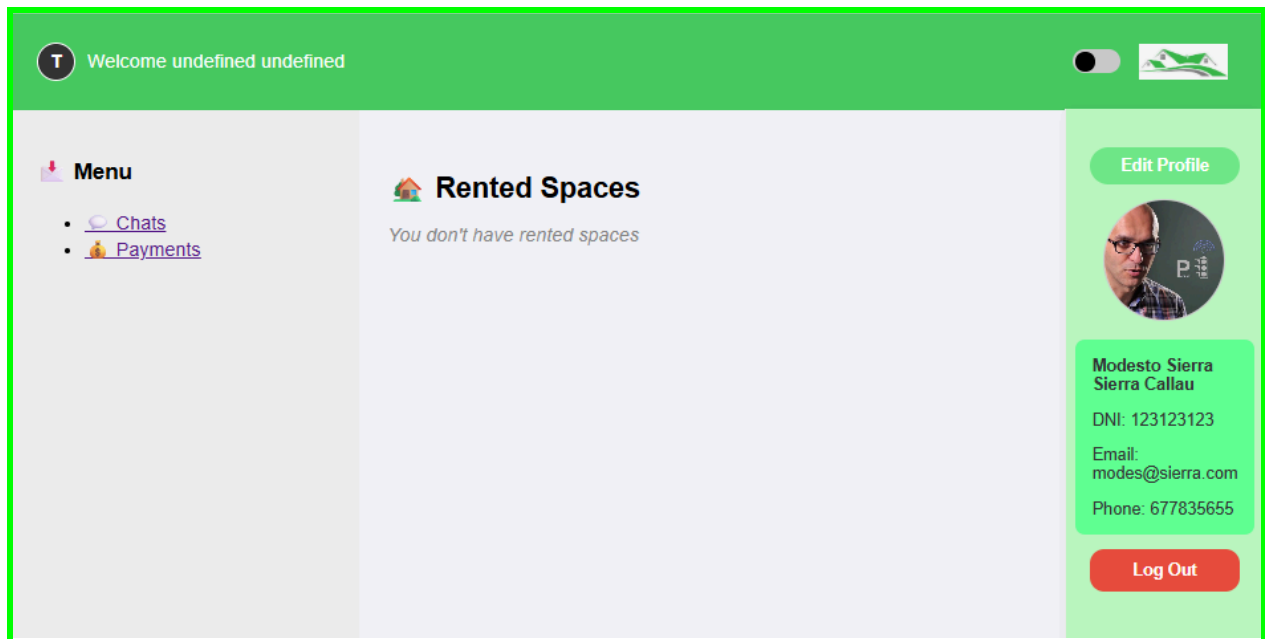


Prueba #003 Cambiar de vista Owner a Tenant

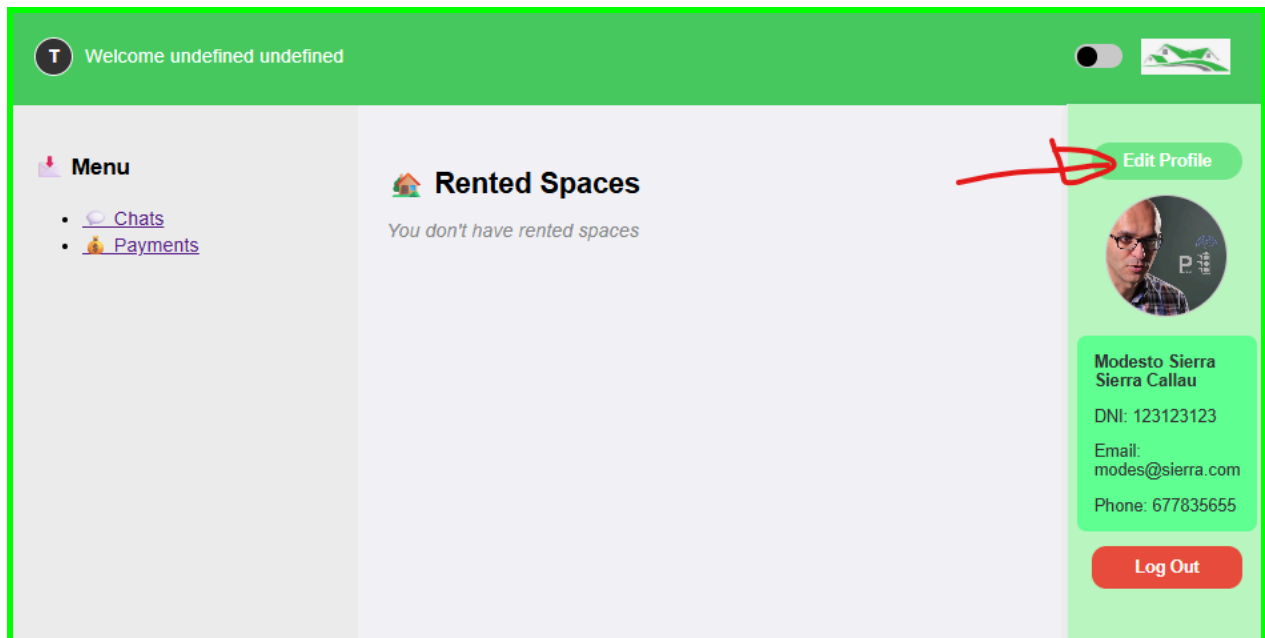
En esta vista verás tus PROPIEDADES porque estás como owner



En esta verás las propiedades en las que eres INQUILINO



Prueba #004 Editar mi perfil



Delete photo

DNI*
123123123

First Name*
Mow

Last Name*
Sierra Sierra Callau

Email*
modes@sierra.com

Phone Number*
677835655

Preference*
Owner

New profile photo

Habr  que adjuntar de nuevo el DNI si cambias tu dni, y esto hasta que no sea validado no se deber  cambiar (No terminado de desarrollar)

DNI card (front)

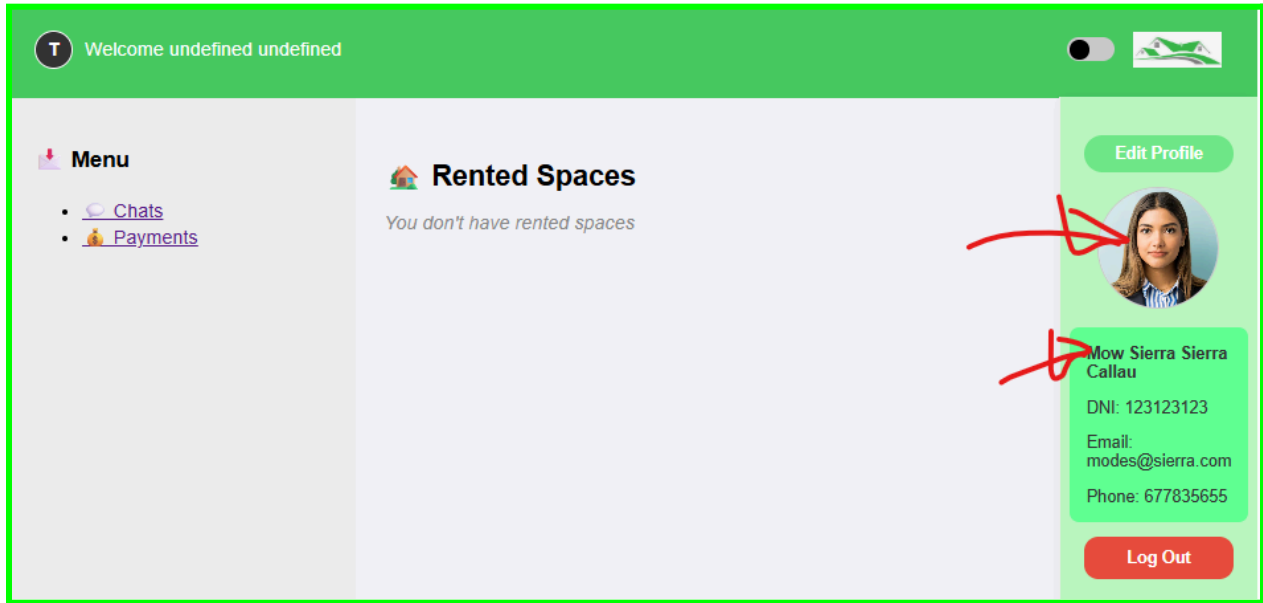
Drag the file or click here

DNI card (back)

Drag the file or click here

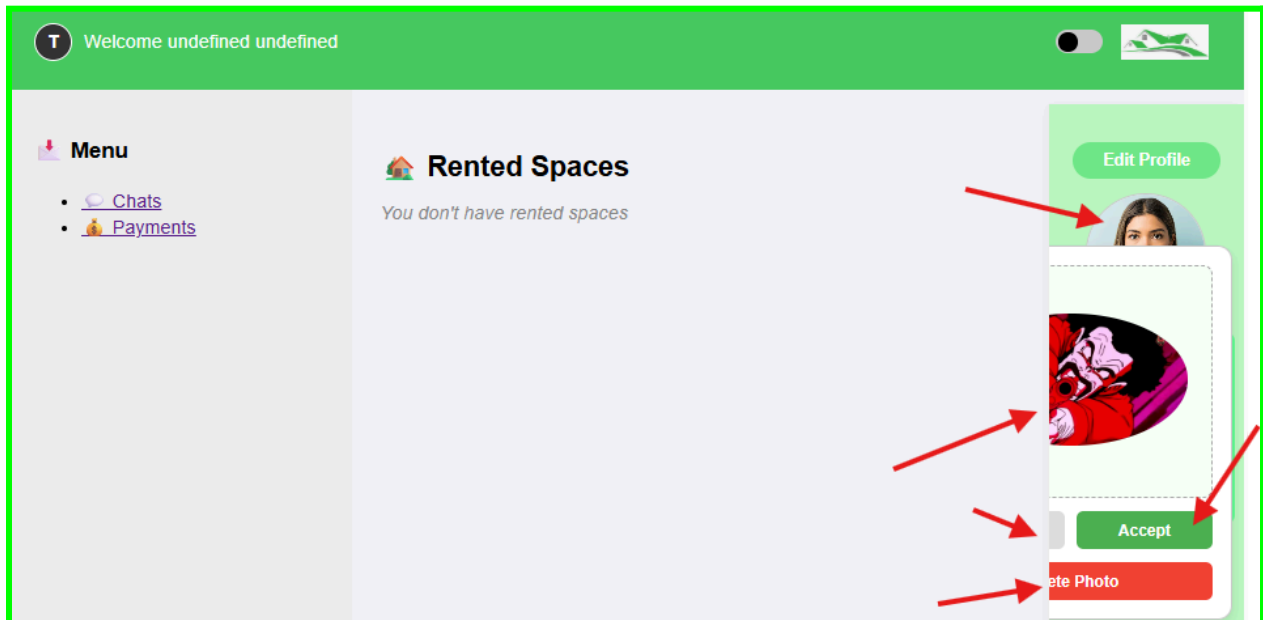
Save changes

[Cancel](#)

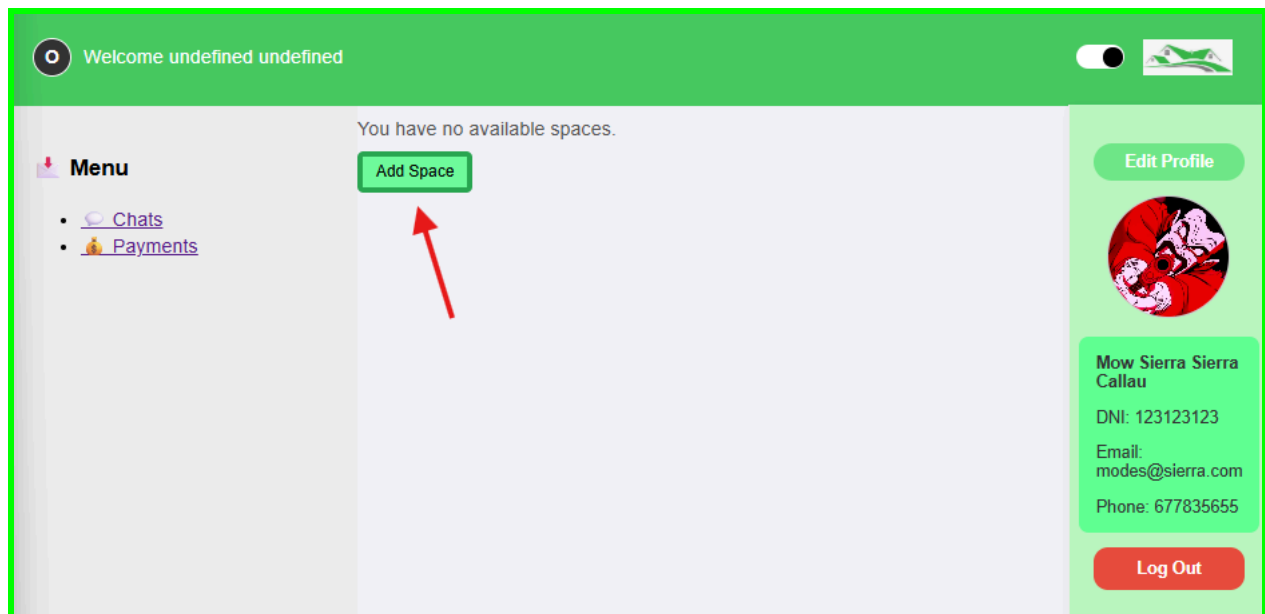


Prueba #004 Cambiar rápido de perfil

No se ve bien por problema de estilo pero es una ventanita donde puedes poner una imagen, aceptar, cancelar o borrar tu perfil actual



Prueba #005 Añadir un Espacio a tu portafolio (SOLO OWNERS)



Rellenamos el formulario

The screenshot shows a form titled "Create New Rental Space". The form has the following fields:

- Space Type: A dropdown menu with "Apartment" selected.
- Space Name: A text input field.
- Floor: A text input field.
- Door: A text input field.
- Square Meters *: A text input field.
- Rooms: A text input field.
- Description: A text area.






Status:

Available

Monthly Price *

Image Gallery:

Drag images or click to select (Max 10)

Property Validation Documents:

Drag documents or click to select (Max 5)

Captura de pantalla
2023-03-27
220526.png

Captura de pantalla
2023-03-29
155257.png

Captura de pantalla
2023-03-30
145719.png

Create Space

Back to Dashboard

Me da un error porque aun no tengo arregla la galeria para que cuando agregue las fotos pueda verse en la vista de propiedades, pero igualmente la creará sin imagenes y repetido

Welcome undefined undefined

Menu

- Chats
- Payments

View details

APARTMENT

Casa antigua de miguel para alquilar

ID: 6828eba785cc142f18b798f3


Price: €480

Status: AVAILABLE

Tenant DNI: Not specified

Validation: PENDING

Edit Profile



Mow Sierra Sierra Callau

DNI: 123123123

Email: modes@sierra.com

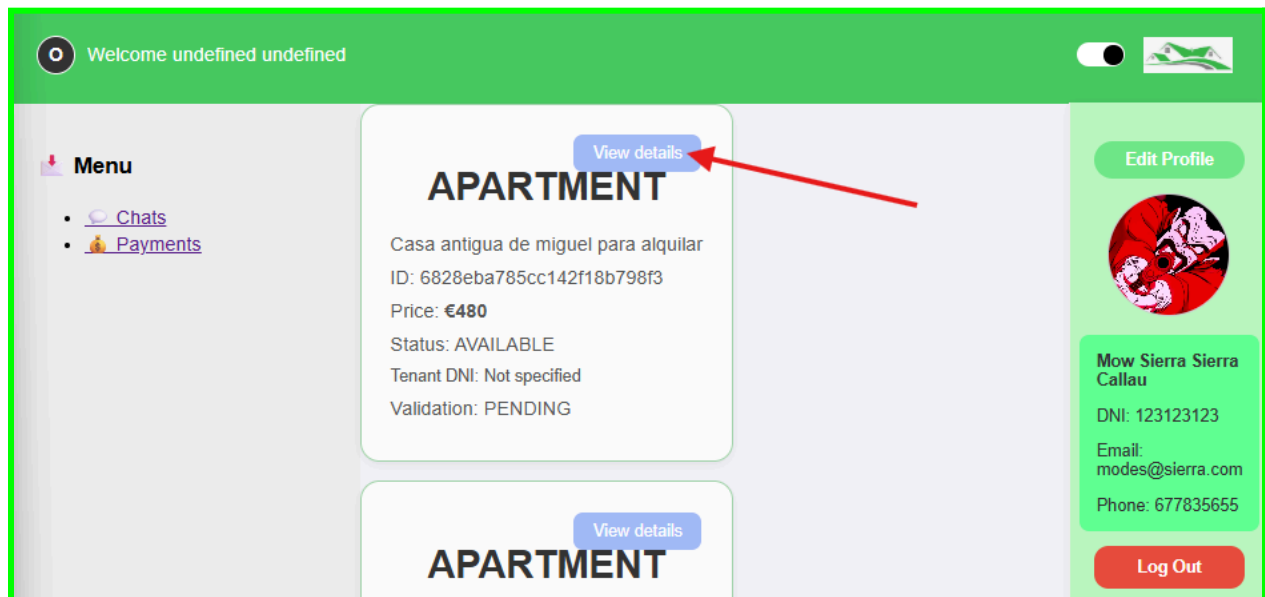
Phone: 677835655

Log Out

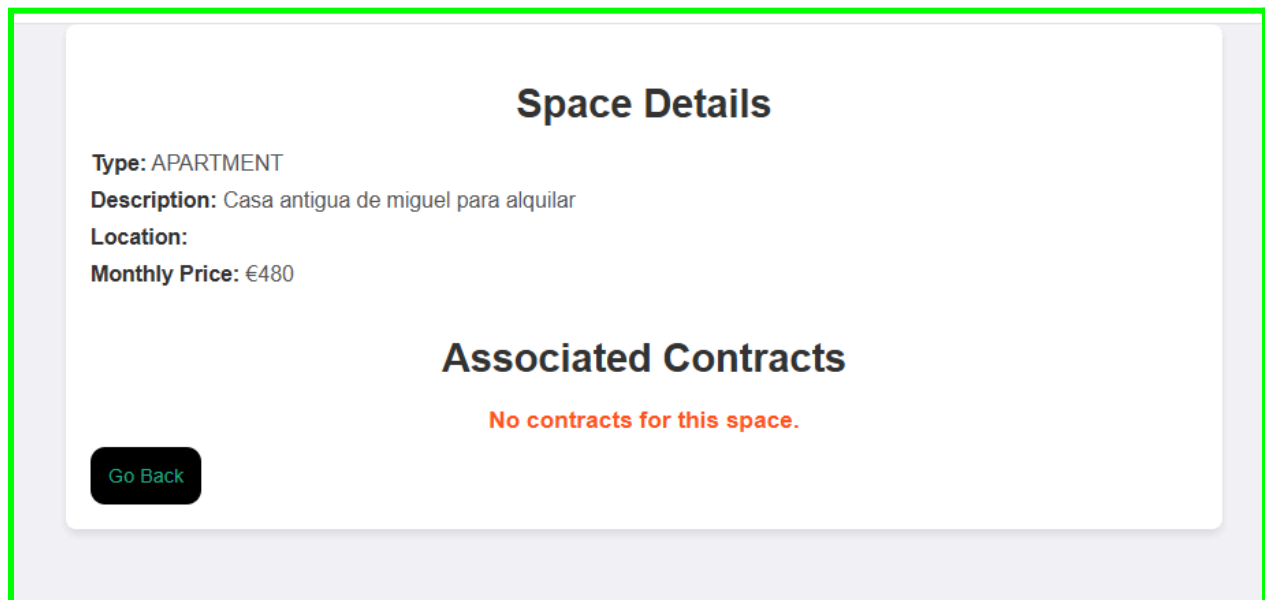
View details

APARTMENT

Prueba #006 Ver detalles de una propiedad

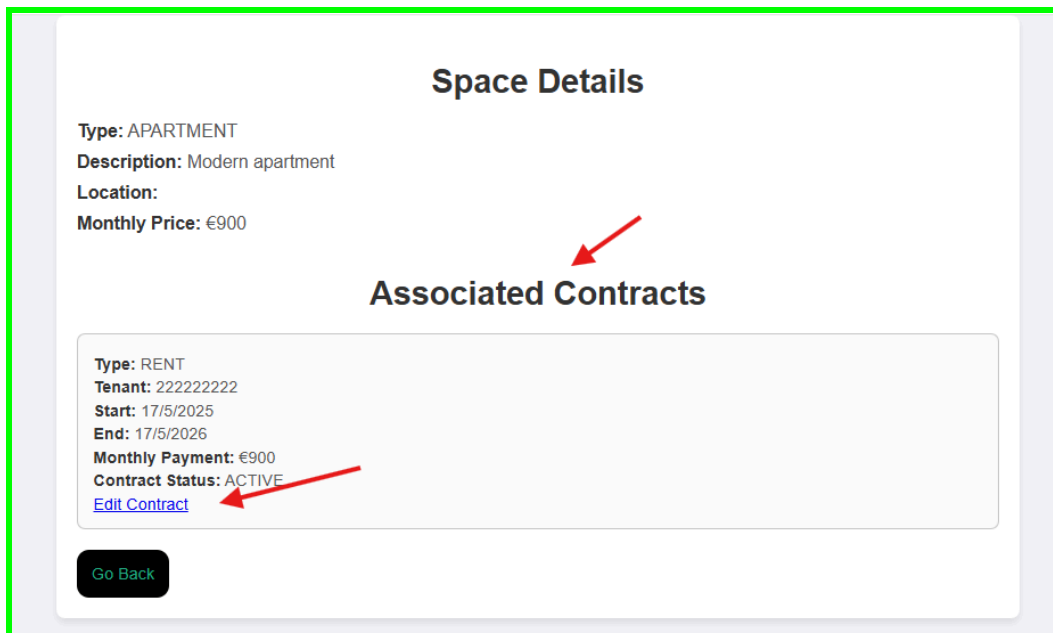


Aqui en esta vista haré que los propietarios puedan agrear contratos (relacionar el espacio con un inquilino)



Prueba #007 Crear contrato a un espacio (SOLO OWNERS)  **No terminada en userinterface**

Prueba #008 Editar un contrato (SOLO OWNERS)  **Me faltan unos toques como el tema de los inquilinos**



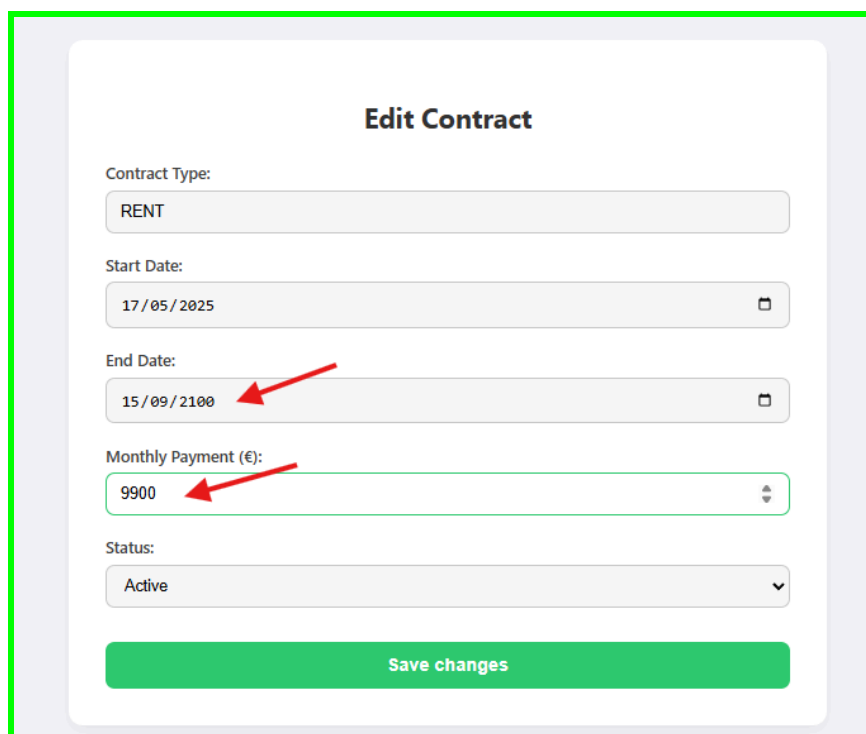
Space Details

Type: APARTMENT
Description: Modern apartment
Location:
Monthly Price: €900

Associated Contracts

Type: RENT
Tenant: 222222222
Start: 17/5/2025
End: 17/5/2026
Monthly Payment: €900
Contract Status: ACTIVE
[Edit Contract](#)

[Go Back](#)



Edit Contract

Contract Type:
RENT

Start Date:
17/05/2025

End Date:
15/09/2100

Monthly Payment (€):
9900

Status:
Active

[Save changes](#)

En principio esta bien, lo haré de modo que un espacio pueda tener múltiples contratos, por cada inquilino, así será flexible

Space Details

Type: APARTMENT
Description: Modern apartment
Location:
Monthly Price: €900

Associated Contracts

Type: RENT
Tenant: 222222222
Start: 17/5/2025
End: 15/9/2100
Monthly Payment: €9900
Contract Status: ACTIVE
[Edit Contract](#)

Go Back

Prueba #009 Ver mis espacios rentados (TENANT)

Esta sería la vista de un inquilino al que le has asociado a un contrato, haré que tengan botones para poder reportar desde aquí mismo el espacio

T

Welcome undefined undefined

Menu

- Chats
- Payments

Rented Spaces


GARAGE - Garage rented
Gallery:
Rooms: Not Available
Monthly Price: €160

View Details

APARTMENT - Flat rented
Gallery:
Rooms: 2
Monthly Price: €800

View Details

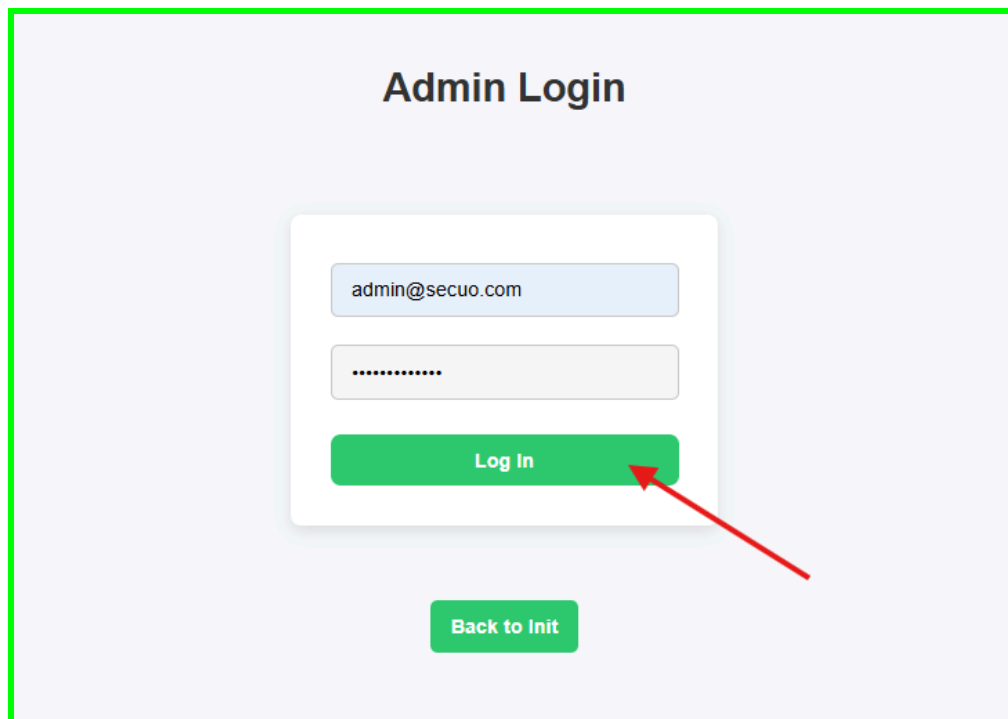
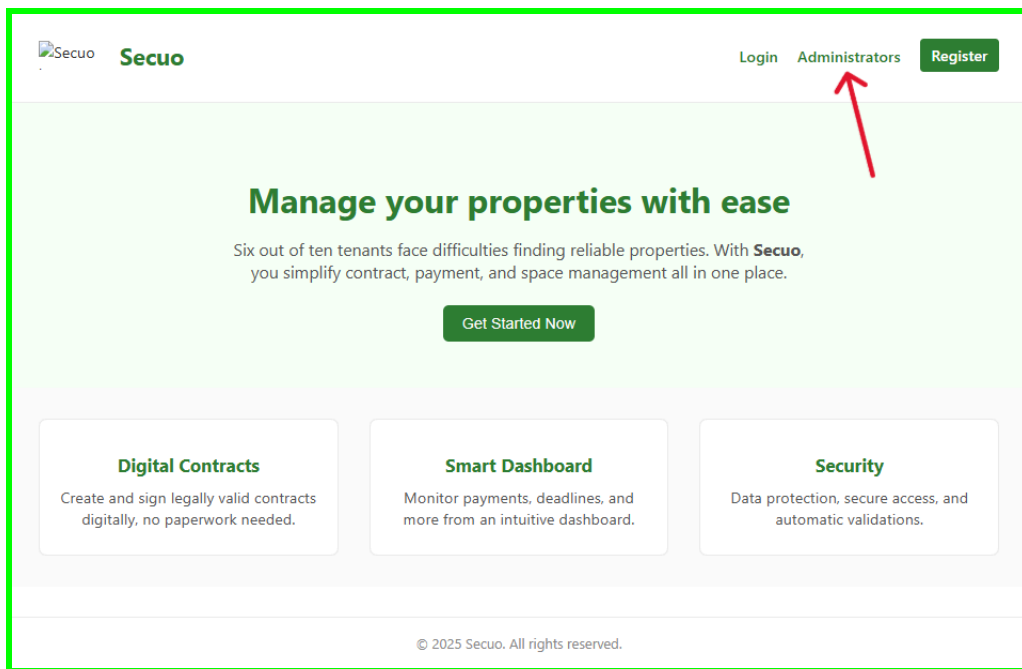
Edit Profile



Oumar Traore
DNI: 000000001
Email: oumar@example.com
Phone: 000111222

Log Out

Prueba #010 Loguearse como administrador



Admin Dashboard

Data Validation

Log Out

En esta vista tendremos las validaciones de usuarios, espacios y contratos, qué es esto?

Cuando se registra un usuario o cuando se crea un contrato o un espacio, se adjuntan unos documentos , como fotos de dni en caso de usuario, o imágenes de certificados de propiedad en caso de las propiedades, o imágenes y documentos de contratos , esto será revisado por los administradores para ver que los documentos que acreditan son válidos, entonces da la vista verde para que el usuario pueda usar el app normal y corrientemente

Data Validation

Pending Users

Juan Perez

Email: juan@example.com | Phone: 123456789

Preference: OWNER | DNI: 111111111

Approve

Reject

Ana Lopez

Email: ana@example.com | Phone: 987654321

Preference: TENANT | DNI: 222222222

Approve

Reject

Carlos Martinez

Email: carlos@example.com | Phone: 567890123

Preference: TENANT | DNI: 333333333

Approve

Reject

Laura Sanchez

Email: laura@example.com | Phone: 123123123

Preference: OWNER | DNI: 444444444

Approve

Reject

Actualmente está simplificado, en un futuro habrá un botón ver que permitirá al administrador ver los documentos adjuntados y compararlos con sus datos introducidos para así validarlos.

Oumar Traore
Email: oumar@example.com | Phone: 000111222
Preference: OWNER | DNI: 000000001

Approve

Reject

Pending Spaces

No pending spaces.

Pending Contracts

RENT - ACTIVE
Owner DNI: 000000001 | Tenant DNI: 222222222
Monthly Payment: €900 | Payment Status: PENDING

Approve

Reject

RENT - ACTIVE
Owner DNI: 000000001 | Tenant DNI: 333333333
Monthly Payment: €150 | Payment Status: PENDING

Approve

Reject

RENT - ACTIVE
Owner DNI: 000000001 | Tenant DNI: 666666666
Monthly Payment: €1100 | Payment Status: PENDING

Approve

Reject

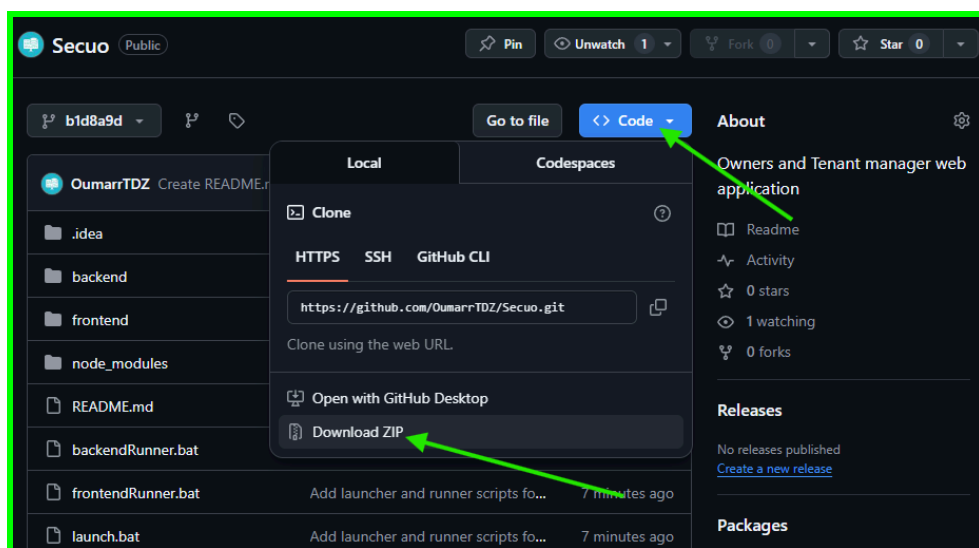
Los usuarios aunque no estén validados podrán usar el app mientras tanto, nuestros admins en pocos minutos revisarán sus documentos y si hay algo mal, se rechazará la solicitud y tendrán que volver a adjuntar datos válidos, hasta que el admin de el visto verde.

10.- MANUAL DE DESPLIEGUE

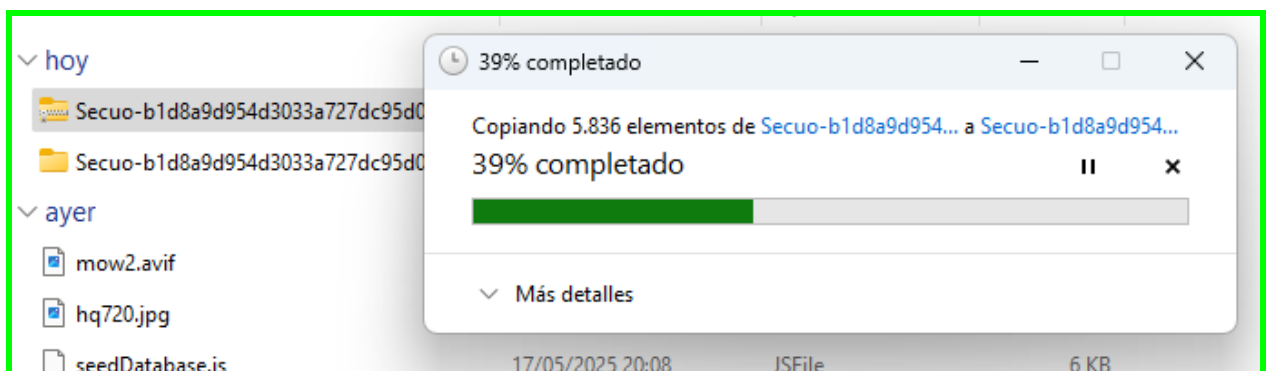
Especificación del procedimiento de instalación y configuración. Puede estar en un anexo

Cuando termine el App ser simplemente acceder a la página web alojado en AWS, por el momento para que puedas acceder lanzaremos la aplicación en local sencillamente gracias a la automatización añadida:

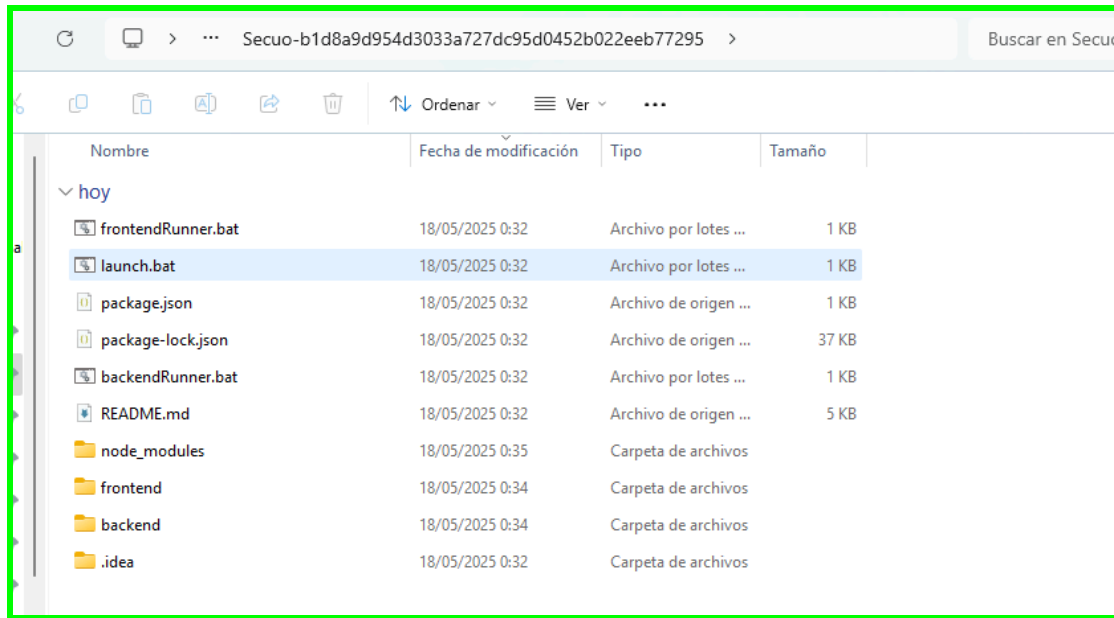
1. Acceda al enlace del repositorio [ENLACE GITHUB](#)
2. Descargue el zip del proyecto (si es usuario avanzado, puede optar por cargar el repositorio en su editor favorito)



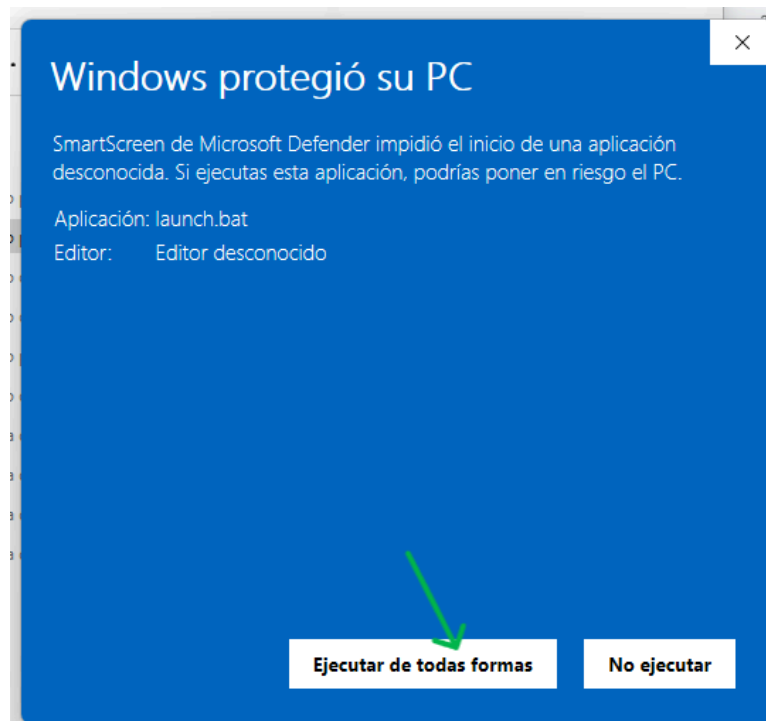
3. Descomprima la carpeta (tomará un poco de tiempo porque tiene los módulos de node)



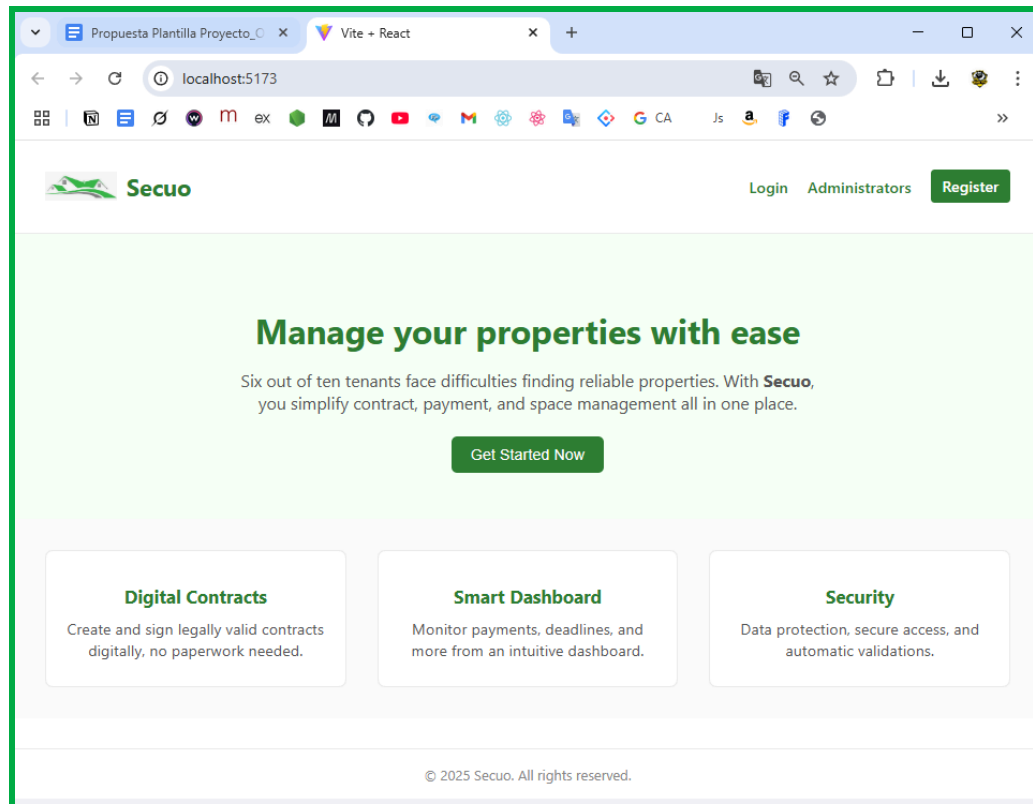
4. Cuando se descomprima por completo, acceda a la carpeta y ejecute el launch.bat (doble-click)



5. Podría darse el caso de que lo bloquee su Windows, le damos a “Mas información” y después a “Ejecutar de todas maneras” (No tenga miedo, si lo desea puede observar la corta fuente del ejecutable, no es ningún troyano)



6. Se iniciarán 3 procesos: Iniciar el backend, cargar la base de datos e iniciar el frontend, posteriormente se abrirá automáticamente nuestra aplicación web y podremos usarla.



7. Mientras pruebas el app en local no cierres ninguna de las 3 consolas porque se usan para levantar el servidor backend y frontend, cuando las cierres tirará el servidor y dejará de funcionar el app local.

En el [backend\seedData.js](#) (que es exactamente el archivo que usamos para cargar la base de datos predefinida) podremos ver usuarios y administradores creados predefinidamente para usar.

Ejemplo de Admin cargado que puedes probar:

email → admin@secuo.com

password → Romingsware8!

Ejemplo de Usuario cargado que puedes probar:

DNI → 000000001

password → 111

DE ESTE PUNTO A ABAJO NO HAY ACTUALIZACIONES.

11.- MANUAL DE USUARIO

Un manual donde se explique cómo utilizar la aplicación por parte de cada perfil de usuario. Puede estar integrado en la propia aplicación, en lugar de estar en este documento o en un documento independiente

Se recomienda incluir entre otros los siguientes apartados:

- Puesta en marcha
- Manejo de los menús o acceso a cada módulo
- Gestión periódica del sistema
- Realización de copias de seguridad
- Mensajes de error
- Glosario de términos

Siempre que sea posible o aplicable se entregará el proyecto con utilidades de instalación que faciliten la implantación del sistema en un nuevo equipo o equipos. Si el proyecto no se ajustara a la creación de una utilidad de instalación se deberá dejar **claramente especificado el procedimiento de instalación o implantación del producto.**

12.- RECURSOS EMPLEADOS

Análisis del desfase entre los recursos planificados y los realmente empleados. Justificación. Se puede considerar incluir alguna tabla y/o un diagrama de Gantt con la diferencia entre el tiempo empleado para cada fase.

13.- ASPECTOS LABORALES Y EMPRESARIALES

Aspectos del proyecto a tratar en este apartado:

1. Presentación del proyecto y el emprendedor: motivos por los que decide emprender y currículum vitae del emprendedor
2. La idea de negocio y la propuesta de valor: se presentan de forma clara y con creatividad.
3. El estudio de mercado analizando el segmento de mercado por criterios de segmentación y analiza los datos del cliente objetivo. Incorpora el lienzo de propuesta de valor (Innokabi).
4. El entorno general: factores de análisis PEST y entorno específico de la empresa. Analiza la competencia. Realiza un análisis DAFO en formato cuadro.
5. Señala cuál es la visión, misión, valores, imagen corporativa y los aspectos de la RSC que va a tener en cuenta en el desarrollo de su actividad. (Responsabilidad Social Corporativa).
6. Las estrategias de marketing estratégico (se posiciona en una de ellas), operativo (establece actuaciones en el marketing mix o 4 p: producto, precio, promoción y distribución).
7. Planificación de los recursos humanos, identificándose con alguna teoría de liderazgo empresarial. Representa la organización de la empresa en un organigrama, realiza un análisis de un puesto de trabajo de su sector y señala las obligaciones laborales de la empresa.
8. Establece la forma jurídica adecuada y los motivos por los que se ha elegido, así como las principales características de la misma.
9. Elabora el plan de producción adaptado a su empresa y señala los costes empresariales clasificándolos en fijos y variables.
10. Indica el valor de las inversiones y de los gastos que vas a realizar para iniciar tu proyecto empresarial.
11. Incluye las fuentes de financiación clasificándolas en propias o ajenas que va a utilizar.

-
12. Análisis contable y financiero: elabora un balance de situación con el formato adecuado y separando los elementos de la empresa en masas patrimoniales.

14.- CONCLUSIONES

Reflexiones y conclusiones a modo de cierre. Por ejemplo, sobre lo aprendido durante el proyecto, dificultades surgidas, qué fase del desarrollo ha gustado más,

15.- BIBLIOGRAFÍA

Referencias bibliográficas o enlaces web que se han consultado a la hora de desarrollar el proyecto.