

Pinares del Norte

Manual Técnico

Por: Bryan Flores

Contents

[Introducción 5](#_Toc197471580)

[Objetivo del Sistema 6](#_Toc197471581)

[Público al que está dirigido 8](#_Toc197471582)

[Alcance del sistema 9](#_Toc197471583)

[Funcionalidades incluidas: 9](#_Toc197471584)

[Funcionalidades fuera del alcance actual: 9](#_Toc197471585)

[Arquitectura del sistema 11](#_Toc197471586)

[Componentes principales: 11](#_Toc197471587)

[1. Frontend (Cliente) 11](#_Toc197471588)

[2. Backend (Servidor) 11](#_Toc197471589)

[3. Base de datos 11](#_Toc197471590)

[4. Sistema de archivos 12](#_Toc197471591)

[Flujo básico del sistema: 12](#_Toc197471592)

[Tecnologías implementadas 13](#_Toc197471593)

[Frontend: 13](#_Toc197471594)

[- Tecnologías base: 13](#_Toc197471595)

[- Librerías y herramientas: 13](#_Toc197471596)

[Backend: 13](#_Toc197471597)

[- Tecnologías base: 13](#_Toc197471598)

[- Librerías y Herramientas: 14](#_Toc197471599)

[Requisitos del sistema (breve) 15](#_Toc197471600)

[Requisitos de hardware: 15](#_Toc197471601)

[Requisitos de software: 15](#_Toc197471602)

[Dependencias 16](#_Toc197471603)

[Instalación 17](#_Toc197471604)

[1. Clona el repositorio 17](#_Toc197471605)

[2. Instala las dependencias necesarias 17](#_Toc197471606)

[3. Base de datos MySQL 17](#_Toc197471607)

[1. Crear base de datos 17](#_Toc197471608)

[2. Configurar conexión en. env 17](#_Toc197471609)

[3. Restaurar la base de datos. 18](#_Toc197471610)

[4. Configuración de la API de Gmail y Contraseña de Aplicación 18](#_Toc197471611)

[Modelo Entidad-Relacion (MER) 20](#_Toc197471612)

[Descripción de las tablas principales 21](#_Toc197471613)

[Organizacion del codigo Fuente 24](#_Toc197471614)

[Módulos o componentes clave 25](#_Toc197471615)

[1. Autenticación (auth/) 25](#_Toc197471616)

[2. Controladores (controllers/) 25](#_Toc197471617)

[3. Configuración (config/) 25](#_Toc197471618)

[4. Vistas (templates/) 25](#_Toc197471619)

[5. Archivos estáticos (static/) 26](#_Toc197471620)

[6. Documentación (docs/) 26](#_Toc197471621)

[7. Rutas globales (routes.py) 26](#_Toc197471622)

[8. Archivo de arranque (run.py) 26](#_Toc197471623)

[11. Principales clases y funciones 27](#_Toc197471624)

[User 27](#_Toc197471625)

[Residences 27](#_Toc197471626)

[Payments 27](#_Toc197471627)

[Claims 27](#_Toc197471628)

[Notifications 27](#_Toc197471629)

[Debts 27](#_Toc197471630)

[Transactions 27](#_Toc197471631)

[Report 28](#_Toc197471632)

[EmailSender 28](#_Toc197471633)

[Mantenimiento y actualizaciones 29](#_Toc197471634)

[Errores comunes y cómo resolverlos 30](#_Toc197471635)

[Endpoints disponibles 31](#_Toc197471636)

[Rutas Generales 31](#_Toc197471637)

[Rutas de Autenticación 31](#_Toc197471638)

[Dashboard Admin 31](#_Toc197471639)

[Gestión de Usuarios (Administrador) 32](#_Toc197471640)

[Gestión de Reclamos (Administrador) 32](#_Toc197471641)

[Pagos (Administrador) 33](#_Toc197471642)

[Actualización de Configuración (Administrador) 33](#_Toc197471643)

[Rutas de Residentes 34](#_Toc197471644)

[Contacto o Soporte 35](#_Toc197471645)

# Introducción

El presente manual técnico tiene como finalidad documentar detalladamente el funcionamiento interno del sistema Pinares del Norte, una plataforma de gestión residencial diseñada para facilitar la administración de pagos, reclamos, reportes, usuarios y apartamentos dentro de un conjunto habitacional.

Este documento está dirigido a desarrolladores, administradores de sistemas y personal técnico que requieran instalar, configurar, mantener o extender el sistema. Aquí se describen los componentes principales del software, sus requerimientos, la estructura del proyecto, y cómo realizar el despliegue y mantenimiento del sistema.

El sistema fue desarrollado utilizando tecnologías como Python con Flask, HTML/CSS/JavaScript para la interfaz, y MySQL para la gestión de datos, siguiendo una arquitectura modular que facilita su comprensión, modificación y escalabilidad.

# Objetivo del Sistema

El objetivo principal de *Habitat Max* es proporcionar una solución digital integral para la gestión eficiente de residenciales. El sistema permite automatizar procesos clave como el registro de pagos mensuales, la administración de usuarios y roles, la gestión de reclamos con evidencias, y el control de ingresos y egresos financieros.

Con este sistema, se busca:

* Optimizar la comunicación entre administradores y residentes.
* Garantizar el registro confiable de transacciones y evidencias.
* Centralizar toda la información relevante de la comunidad en un solo lugar.

Pinares del Norte está diseñado para ser escalable, intuitivo y seguro, permitiendo su uso tanto en residenciales pequeños como en complejos habitacionales más grandes.

# Público al que está dirigido

El sistema *Pinares del Norte* está dirigido principalmente a dos tipos de usuarios:

Administradores del residencial:

* Encargados de gestionar pagos, controlar el historial financiero, registrar egresos, resolver reclamos, y mantener actualizada la información de los residentes y sus viviendas.
* Utilizan el panel administrativo del sistema, el cual incluye herramientas avanzadas de control y reportes.

Residentes:

* Propietarios o inquilinos de las viviendas que forman parte del residencial.
* Acceden al panel de usuario para consultar deudas, subir evidencias de pagos, ver el historial de transacciones, y recibir notificaciones importantes.

Este sistema también puede ser utilizado por desarrolladores y técnicos encargados del soporte o mantenimiento de la aplicación, quienes requerirán acceso al manual técnico para entender su arquitectura y funcionamiento interno.

# Alcance del sistema

El sistema *Pinares del Norte* abarca las funcionalidades necesarias para administrar de forma eficiente un conjunto residencial. Su alcance se limita a las operaciones administrativas internas entre los residentes y los administradores del residencial.

## Funcionalidades incluidas:

Registro y gestión de usuarios con roles diferenciados (administrador y residente).

Asignación y liberación de apartamentos.

Registro manual de pagos por parte del administrador.

Envío de evidencias de pago por parte de los residentes.

Aprobación de pagos enviados con evidencia.

Generación automática de deuda mensual por vivienda.

Registro de egresos con motivo y responsable.

Panel de usuario para residentes con historial de pagos y notificaciones.

Gestión de reclamos con carga de evidencias y solución programada.

Asignación de edificios y casas.

## Funcionalidades fuera del alcance actual:

Integración con plataformas de pago en línea (como Stripe o PayPal).

Gestión de seguridad o vigilancia del residencial.

Gestión de visitantes o proveedores externos.

Aplicación móvil (actualmente solo disponible como aplicación web).

Envío automático de notificaciones por SMS (solo notificaciones internas).

# Arquitectura del sistema

El sistema *Pinares del Norte* está construido bajo una arquitectura cliente-servidor de tipo web, utilizando el patrón Modelo-Vista-Controlador (MVC) para mantener la separación de responsabilidades y facilitar el mantenimiento y escalabilidad.

## Componentes principales:

### 1. Frontend (Cliente)

* Desarrollado con HTML, Tailwind CSS, JavaScript, Alpine.js y bibliotecas como Flowbite y DaisyUI.
* Responsivo y adaptado a múltiples dispositivos.
* Se comunica con el backend mediante formularios y rutas HTTP (GET, POST).
* Usa plantillas Jinja2 para renderizar contenido dinámico.

### 2. Backend (Servidor)

* Construido con Flask, un microframework de Python.
* Se encarga de gestionar la lógica de negocio, validaciones, sesiones de usuario y conexión con la base de datos.
* Organizado en controladores (routes), modelos (consultas SQL y lógica de datos) y plantillas (vistas renderizadas).

### 3. Base de datos

* Usa MySQL como sistema de gestión.
* Estructurada en múltiples tablas relacionadas entre sí, tales como: users, roles, transactions, claims, apartments, buildings, entre otras.
* Relación uno a uno (por ejemplo, usuario-apartamento) y muchos a muchos (usuario-roles).

### 4. Sistema de archivos

* Las imágenes de perfil y evidencias se almacenan localmente en carpetas organizadas por tipo (uploads/profile\_pics, uploads/evidences/).
* Las evidencias de reclamos se guardan con el formato claim\_id\_fileN en subcarpetas por usuario.

### Flujo básico del sistema:

1. El usuario accede al sistema desde su navegador.
2. El frontend envía una solicitud HTTP al backend.
3. Flask procesa la solicitud, accede a la base de datos si es necesario y genera la respuesta.
4. La vista se renderiza con datos dinámicos mediante Jinja2 y se envía al navegador del usuario.

# Tecnologías implementadas

## Frontend:

### - Tecnologías base:

- HTML

- CSS

- JavaScript

### - Librerías y herramientas:

- Tailwind CSS

- Flowbite

- DaisyUI

- Google Fonts

- Google Icons

- Heroicons

- SailboatUI

- MerakiUI

- Alpine.js

- Undraw

- Jinja2

## Backend:

### - Tecnologías base:

- Flask (Python)

- MySQL

### - Librerías y Herramientas:

- Bcrypt

- Mysql-connector

- Gmail (API)

# Requisitos del sistema (breve)

## Requisitos de hardware:

* Procesador: Dual-Core 2.0 GHz o superior
* Memoria RAM: 4 GB mínimo
* Almacenamiento: 10 GB disponibles
* Conexión a internet o red local

## Requisitos de software:

* Sistema operativo: Windows 10 / Ubuntu 20.04
* Python 3.10 o superior
* MySQL 8.x
* Navegador moderno (Chrome, Firefox, etc.)
* Framework Flask

# Dependencias

APScheduler==3.11.0

bcrypt==4.2.1

email\_validator==2.2.0

Flask==3.0.3

flask-cors==5.0.1

Flask-SQLAlchemy==3.1.1

Jinja2==3.1.4

MarkupSafe==2.1.5

mysqlclient==2.2.5

pandas==2.2.3

python-dotenv==1.0.1

qrcode==8.0

reportlab==4.3.0

requests==2.32.3

Werkzeug==3.0.3

yagmail==0.15.293

premailer==3.10.0

# Instalación

Asegúrate de tener \*\*Python 3.10 o superior\*\* y \*\*pip\*\* correctamente instalados.

## 1. Clona el repositorio

git clone https://github.com/brxnzy/Residential-Management.git

cd Residential-Management

## 2. Instala las dependencias necesarias

pip install -r requirements.txt

## 3. Base de datos MySQL

### 1. Crear base de datos

Abre tu gestor de base de datos MySQL y crea una base de datos. y asegurate de copiar el script que está en el archivo SQL que está dentro de app que contiene todas las tablas.

CREATE DATABASE pinares\_del\_norte;

### 2. Configurar conexión en. env

Crea un archivo llamado `. env` en la raíz del proyecto con el siguiente contenido:

DB\_HOST=localhost

DB\_USER=tu\_usuario

DB\_PASSWORD=tu\_contraseña

DB\_NAME=pinares\_del\_norte

Reemplaza `tu\_usuario` y `tu\_contraseña` con tus datos reales de conexión MySQL.

### 3. Restaurar la base de datos.

el archivo que se llama db.sql restauralo en MySQL Workbench ya que contiene la base de datos completa de la aplicacion.

### 4. Configuración de la API de Gmail y Contraseña de Aplicación

Si planeas utilizar la API de Gmail o enviar correos electrónicos desde tu cuenta de Gmail, necesitarás configurar una \*\*contraseña de aplicación\*\* para tu cuenta de Google. Aquí están los pasos para hacerlo:

1. Habilita la verificación en dos pasos para tu cuenta de Google si aún no lo has hecho.

2. Genera una contraseña de aplicación siguiendo estos pasos:

3. dirígete a tu cuenta de Google: [https://myaccount.google.com](https://myaccount.google.com).

En el menú de la izquierda, haz clic en Seguridad

Desplázate hacia abajo hasta la sección Acceso a Google y selecciona Contraseñas de aplicaciones.

\* Selecciona el tipo de dispositivo (puede ser "Otro" si no aparece un dispositivo específico) y luego haz clic en Generar.

Guarda la contraseña generada, ya que la necesitarás más tarde.

con el .env ya creado en la raíz del proyecto y agrega las siguientes variables de entorno:

EMAILUSER="tu\_email@gmail.com"

EMAILPASSW="tu\_contraseña\_de\_aplicación"

Reemplaza:

`tu\_email@gmail.com` con tu dirección de correo electrónico de Gmail.

`tu\_contraseña\_de\_aplicación` con la contraseña de aplicación generada en los pasos anteriores.

Nota: La contraseña de aplicación no es la misma que tu contraseña normal de Gmail. Es una contraseña única que se genera específicamente para su uso en aplicaciones externas.

# Modelo Entidad-Relacion (MER)

# Descripción de las tablas principales

A continuación, se describen las tablas principales utilizadas en la base de datos del sistema:

1. users

Almacena la información de los usuarios del sistema (residentes y administradores).  
Campos: id, id\_card,name, last\_name, password, email, status, photo, registration\_date, reset\_token

2. roles

Define los diferentes roles que un usuario puede tener (por ejemplo: residente, administrador).  
Campos: id, name

3. user\_roles

Tabla intermedia para la relación muchos a muchos entre usuarios y roles.  
Campos: user\_id, role\_id

4. apartments

Registra los apartamentos y su asignación a usuarios.  
Campos clave: id, building, apartment\_number, id\_usuario(fk), occupied

5. houses

Registra las casas y su asignación a usuariosdentro de una estructura habitacional.  
Campos clave: id, house\_number, id\_usuario, occupied

6. payments

Historial de pagos confirmados, ya sea en efectivo o por transferencia.  
Campos clave: id, user\_id, amount, method, date

7. claims

Registra los reclamos realizados por los usuarios, incluyendo evidencia.  
Campos clave: id, id\_usuario, category, description, status, created\_at, evidence\_1, evidence\_2, evidence\_3, scheduled\_date, start\_time, end\_time, finished\_at, rating

8. notifications

Almacena notificaciones que serán mostradas a los usuarios.  
Campos clave: id, id\_usuario, message, status, reply

9. transfer\_requests

Solicitudes de pagos realizadas por transferencia.  
Campos clave: id, id\_usuario, evidence, status, created\_at, description, debt\_id

10. debts

Registra las deudas de los residentes.  
Campos clave: id, id\_usuario, period, month, amount, created\_at

11. transfer\_request\_debts

Tabla intermedia que asocia solicitudes de transferencia con una o más deudas.  
Campos clave: transfer\_request\_id, debt\_id

12. debt\_amount

Registra el monto correspondiente a las deudas que se generaran al crear los residentes o mensualmente período.  
Campos clave:amount

13. transactions

Historial financiero general del sistema, tanto ingresos como egresos.  
Campos clave: id, amount, type (ingreso/egreso), admin\_id, reason, created\_at

# Organizacion del codigo Fuente

La estructura del proyecto *Pinares del Norte* se basa en el patrón de arquitectura MVC (Modelo - Vista - Controlador), que permite una separación clara de responsabilidades para mejorar la mantenibilidad y escalabilidad del sistema.

A screenshot of a computer

AI-generated content may be incorrect.

# Módulos o componentes clave

El sistema *Pinares del Norte* se divide en varios módulos fundamentales que permiten su correcto funcionamiento y organización. A continuación, se describen los componentes más importantes:

## 1. Autenticación (auth/)

Gestiona el inicio de sesión, cierre de sesión, validación de usuarios, y asignación de roles (residente o administrador).

## 2. Controladores (controllers/)

Contiene la lógica principal del sistema, incluyendo el manejo de rutas y funcionalidades como pagos, reclamos, notificaciones, usuarios, apartamentos y transacciones.

## 3. Configuración (config/)

Define los parámetros globales del sistema, como conexión a la base de datos, claves secretas y configuración de entorno.

## 4. Vistas (templates/)

Incluye las interfaces de usuario divididas por tipo de usuario:

* admin/: Panel de control y funciones administrativas.
* resident/: Panel de usuario para residentes.
* secure/: Vistas seguras para recuperación o establecimiento de contraseña.
* errors/: Páginas de error personalizadas.
* includes/: Fragmentos reutilizables.

## 5. Archivos estáticos (static/)

Contiene los recursos del frontend (CSS, JS, imágenes, íconos, fuentes y archivos subidos).

## 6. Documentación (docs/)

Ubicación de archivos del manual técnico y otros documentos internos del proyecto.

## 7. Rutas globales (routes.py)

Archivo donde se agrupan y registran las rutas principales del sistema, enlazando los controladores correspondientes.

## 8. Archivo de arranque (run.py)

Archivo encargado de iniciar la aplicación con Flask

# 11. Principales clases y funciones

## User

Administra usuarios: crear, editar, eliminar, asignar roles y apartamentos.

## Residences

Gestiona casas y apartamentos: obtener disponibles, asignar y desalojar.

## Payments

Registra pagos en efectivo y por transferencia; acepta o rechaza solicitudes; obtiene historial de pagos.

## Claims

Envía, atiende, rechaza y califica reclamos; permite programar solución.

## Notifications

Envía recordatorios y avisos; permite marcar como leído y consultar notificaciones.

## Debts

Genera y actualiza deudas; obtiene deudores y modifica montos.

## Transactions

Registra egresos; obtiene ingresos, gastos y estadísticas financieras.

## Report

Genera facturas en PDF de los pagos realizados (efectivo o transferencia), incluyendo detalles como monto, fecha, método y nombre del residente.

## EmailSender

Envía correos electrónicos para:

* Activación de cuenta con enlace personalizado.
* Recuperación de contraseña.
* Envío automático de factura al correo tras confirmar una transferencia.

# Mantenimiento y actualizaciones

El sistema *Pinares del Norte* requiere tareas periódicas para garantizar su buen funcionamiento, seguridad y escalabilidad. Estas incluyen:

Mantenimiento preventivo

* Revisión de logs de errores.
* Limpieza de archivos temporales y evidencias no vinculadas.
* Verificación del espacio de almacenamiento (especialmente en uploads/).
* Revisión de accesos no autorizados o inusuales.

Actualizaciones

* Incorporación de nuevas funcionalidades según las necesidades del residencial.
* Mejoras en el diseño y experiencia del usuario.
* Actualización de dependencias y librerías del proyecto.
* Ajustes en el cálculo de estadísticas, generación de deudas o validaciones adicionales.

Recomendaciones

* Realizar copias de seguridad semanales de la base de datos.
* Documentar cada cambio o actualización en un archivo CHANGELOG.md.
* Probar cada nueva función en entorno local antes de subir a producción.

# Errores comunes y cómo resolverlos

A continuación, se describen algunos errores frecuentes en el sistema *Pinares del Norte* y sus respectivas soluciones:

Usuario no encontrado  
Esto ocurre cuando se intenta iniciar sesión con un correo que no existe o que fue eliminado. Verifica que el correo esté bien escrito y que el usuario exista en la base de datos.

"No se puede registrar pago, el usuario no tiene deuda"  
El sistema solo permite registrar pagos si hay una deuda activa. Asegúrate de consultar primero las deudas antes de registrar un pago manualmente.

“Archivo no permitido" al subir evidencia  
El sistema rechaza archivos con extensiones no válidas. Solo se aceptan imágenes y archivos PDF, como .jpg, .png o .pdf. Evita archivos ejecutables.

Correo no enviado  
Este error indica un fallo en el envío de correos electrónicos. Puede deberse a una mala configuración del servidor SMTP en el archivo. env ,falta de conexión a internet o un correo no existente o erróneo. Revisa los parámetros y realiza pruebas de conexión.

# Endpoints disponibles

## Rutas Generales

* GET /static/<path:filename>: Ruta para acceder a archivos estáticos.
* GET /routes: Lista las rutas disponibles en la aplicación.
* GET /disabled: Muestra las rutas de usuarios deshabilitados.
* GET /logout: Cierra la sesión de un usuario.
* GET /: Redirige al dashboard.

## Rutas de Autenticación

* GET, POST /login: Ruta para iniciar sesión.
* GET, POST /forgot\_ur\_password: Ruta para solicitar recuperación de contraseña.
* GET, POST /reset\_password/<token>: Ruta para resetear la contraseña usando un token.
* GET, POST /activate\_account/<int:user\_id>: Ruta para activar una cuenta.

## Dashboard Admin

* GET /admin: Acceso al panel administrativo.
* GET /admin/<section>/: Acceso a secciones específicas dentro del panel administrativo.
* GET /admin/<section>/<sub\_section>: Acceso a subsecciones dentro del panel administrativo.

## Gestión de Usuarios (Administrador)

* POST /admin/add\_user: Agregar un nuevo usuario.
* POST /admin/delete\_user/<int:user\_id>: Eliminar un usuario por ID.
* POST /admin/disable\_user: Deshabilitar un usuario.
* POST /admin/enable\_user/<int:user\_id>: Habilitar un usuario.
* POST /admin/update\_roles/<int:user\_id>: Actualizar los roles de un usuario.
* POST /admin/assign\_property/<int:user\_id>: Asignar propiedad a un usuario.
* GET /admin/user\_info/<int:user\_id>: Obtener información de un usuario.
* POST /admin/residences/vacate\_residence: Vaciar una residencia de su usuario.

## Gestión de Reclamos (Administrador)

* POST /admin/attend\_claim: Asistir un reclamo.
* POST /admin/finish\_claim: Terminar un reclamo.
* POST /admin/reject\_claim: Rechazar un reclamo.

## Pagos (Administrador)

* POST /admin/payments/cash\_payment: Realizar un pago en efectivo.
* GET /admin/payment\_complete: Confirmar la finalización de un pago.
* GET /admin/payments/user\_debts/<int:user\_id>: Ver la deuda de un usuario.
* POST /admin/payments/accept\_transfer\_request: Aceptar una solicitud de transferencia.
* POST /admin/payments/reject\_transfer\_request: Rechazar una solicitud de transferencia.
* POST /admin/register\_expense: Registrar un egreso o gasto.
* POST /admin/payments/payment-reminder: Enviar recordatorio de pago.

## Actualización de Configuración (Administrador)

* POST /admin/update\_logo: Actualizar el logo.
* POST /admin/update\_debt\_amount: Actualizar el monto de deuda.
* POST /admin/update\_photo: Actualizar foto de perfil.
* POST /admin/delete\_photo: Eliminar foto de perfil.
* POST /admin/update\_profile: Actualizar información de perfil.
* POST /admin/update\_password: Cambiar contraseña.

## Rutas de Residentes

* GET /resident: Página principal del residente.
* POST /resident/update\_photo/<int:user\_id>: Actualizar la foto de perfil de un residente.
* POST /resident/delete\_photo/<int:user\_id>: Eliminar la foto de perfil de un residente.
* POST /resident/update\_info/<int:user\_id>: Actualizar información del residente.
* POST /resident/update\_password/<int:user\_id>: Actualizar contraseña del residente.
* POST /resident/send\_claim: Enviar un reclamo desde el residente.
* POST /resident/rate\_claim: Valorar un reclamo.
* POST /resident/mark\_as\_read: Marcar un reclamo como leído.
* POST /resident/transfer\_request: Solicitar una transferencia de propiedad.

# Contacto o Soporte

Si necesitas ayuda adicional, puedes comunicarte con el equipo de soporte de Pinares del Norte través de los siguientes medios:

* Correo electrónico: respinaresdelnorte@gmail.com
* Teléfono: +1 (849-284-2781)

Nota: Para reportar errores o fallos técnicos, por favor incluye una descripción clara del problema, la fecha y hora en que ocurrió, y, si es posible, una captura de pantalla.