

Documentación

Parqueos Callejeros

Carrera: Ingeniería en Computación

Curso: Taller de Programación

Autor: Santiago Villarreal Arley

Carné: 2025120897

Profesor: William Mata

Fecha de entrega: 3 de junio 2025

**Tabla de contenidos**

[1. Enunciado del proyecto 5](#_Toc199688403)

[Objetivo General 5](#_Toc199688404)

[Definición del Proyecto 5](#_Toc199688405)

[Requerimientos Funcionales 5](#_Toc199688406)

[Requerimientos Técnicos 5](#_Toc199688407)

[2. Temas Investigados 7](#_Toc199688408)

[1. Interfaces gráficas con tkinter 7](#_Toc199688409)

[Marco teórico 7](#_Toc199688410)

[Fuentes: 7](#_Toc199688411)

[2. Archivos JSON como base de datos 7](#_Toc199688412)

[Marco teórico: 7](#_Toc199688413)

[Aplicación en el proyecto: 7](#_Toc199688414)

[Fuentes: 7](#_Toc199688415)

[3. Gestión de múltiples pantallas (Frame) 7](#_Toc199688416)

[Marco teórico: 7](#_Toc199688417)

[Aplicación en el proyecto: 7](#_Toc199688418)

[Fuentes: 8](#_Toc199688419)

[4. Validación de fechas y tiempos con datetime 8](#_Toc199688420)

[Marco teórico: 8](#_Toc199688421)

[Aplicación en el proyecto: 8](#_Toc199688422)

[5. Envío de correos con smtplib 8](#_Toc199688423)

[Marco teórico: 8](#_Toc199688424)

[Aplicación en el proyecto: 8](#_Toc199688425)

[6. Generación de PDF con reportlab 9](#_Toc199688426)

[Marco teórico: 9](#_Toc199688427)

[Aplicación en el proyecto: 9](#_Toc199688428)

[Fuentes: 9](#_Toc199688429)

[7. Uso de ttk.Treeview 9](#_Toc199688430)

[Marco teórico: 9](#_Toc199688431)

[Aplicación en el proyecto: 9](#_Toc199688432)

[Fuentes: 10](#_Toc199688433)

[8. Cifrado de contraseñas con hashlib 10](#_Toc199688434)

[Marco teórico: 10](#_Toc199688435)

[Aplicación en el proyecto: 10](#_Toc199688436)

[Fuentes: 10](#_Toc199688437)

[9. División en módulos reutilizables 10](#_Toc199688438)

[Marco teórico: 10](#_Toc199688439)

[Aplicación en el proyecto: 10](#_Toc199688440)

[Fuentes: 10](#_Toc199688441)

[Conclusiones del trabajo 11](#_Toc199688442)

[Problemas encontrados y soluciones aplicadas 11](#_Toc199688443)

[Cambio de estructura del archivo pc\_espacios.json 11](#_Toc199688444)

[Envío de correos en entornos sin conexión o sin acceso SMTP 11](#_Toc199688445)

[Validación de alquileres vencidos 11](#_Toc199688446)

[Renderizado de múltiples marcos en Tkinter 11](#_Toc199688447)

[Adaptación de reportes a PDF 11](#_Toc199688448)

[Aprendizajes obtenidos 12](#_Toc199688449)

[Dominio de tkinter a nivel intermedio-avanzado 12](#_Toc199688450)

[Uso profesional de JSON como almacenamiento 12](#_Toc199688451)

[Implementación modular escalable 12](#_Toc199688452)

[Integración de tecnologías externas 12](#_Toc199688453)

[Aplicación práctica del desarrollo iterativo 12](#_Toc199688454)

[Estadística de tiempos invertidos 12](#_Toc199688455)

[Lista de revision del Proyecto 13](#_Toc199688456)

[Referencias Bibliográficas 16](#_Toc199688457)

# 1. Enunciado del proyecto

## Objetivo General

Desarrollar un sistema para la gestión de parqueos callejeros que permita a los usuarios registrar alquileres de espacios, a los administradores configurar el sistema y controlar los espacios disponibles, y a los inspectores verificar el cumplimiento del pago de parqueo en tiempo real.

## Definición del Proyecto

Este sistema permite el control de espacios públicos de parqueo mediante una plataforma construida en Python, con interfaz gráfica (GUI) en Tkinter, y persistencia de datos mediante archivos JSON. Se divide en tres aplicaciones principales:

* **Usuarios**: Registro, parqueo, agregar tiempo, desaparcar, reportes.
* **Administradores**: Configuración del parqueo, gestión de espacios, reportes.
* **Inspectores**: Revisión de parqueos y generación de multas, reportes.

## Requerimientos Funcionales

* Módulo de configuración del parqueo (horario, precios, multas).
* Módulo para registrar espacios de parqueo y su estado.
* Gestión de usuarios, autenticación, recuperación y cifrado de contraseñas.
* Funcionalidad para alquilar, agregar tiempo, desaparcar y ver reportes.
* Sistema de inspección y generación automática de multas.
* Reportes en formato PDF, envío de correos automáticos.
* Manual de usuario y opción de ayuda.

## Requerimientos Técnicos

* Lenguaje: Python 3.12+
* GUI: Tkinter
* Persistencia: Archivos JSON (pc\_configuracion.json, pc\_espacios.json, pc\_usuarios.json, pc\_alquileres.json, pc\_multas.json)
* Documentación interna en código
* Estructura modular dividida en carpetas (frames, módulos, data)

# 2. Temas Investigados

## 1. Interfaces gráficas con tkinter

## Marco teórico

Tkinter es la biblioteca estándar de interfaces gráficas para Python. Proporciona elementos como ventanas, botones, menús y cuadros de texto, que permiten construir GUIs (interfaces gráficas de usuario) multiplataforma.  
Aplicación en el proyecto:  
Todo el sistema (usuario, administrador e inspector) fue construido con tkinter, incluyendo la navegación entre pantallas (frames), validación de entrada, menús, botones y vistas con ttk.Treeview.

### Fuentes:

Python Docs: [https://docs.python.org/3/library/tkinter.html](https://docs.python.org/3/library/tkinter.html" \t "_new)

## 2. Archivos JSON como base de datos

## Marco teórico:

El módulo json de Python permite codificar y decodificar estructuras de datos a y desde el formato JSON, que es ampliamente usado para persistencia ligera.

### Aplicación en el proyecto:

Todos los datos se almacenan como archivos .json, entre ellos: pc\_usuarios.json, pc\_espacios.json, pc\_multas.json, pc\_configuracion.json, etc. Las funciones de lectura y escritura están encapsuladas en modulo\_utiles.py.

## Fuentes:

Python JSON: [https://docs.python.org/3/library/json.html](https://docs.python.org/3/library/json.html" \t "_new)

## 3. Gestión de múltiples pantallas (Frame)

Marco teórico:  
Tkinter permite dividir la interfaz en pantallas usando clases que heredan de tk.Frame. Para simular navegación, se destruye el Frame actual y se instancia el nuevo.

Aplicación en el proyecto:  
App.cambiar\_frame() se usa para mostrar dinámicamente diferentes pantallas según el rol (usuario, administrador, inspector).

### Fuentes:

* Tkinter Switching Frames: [https://stackoverflow.com/a/7557028](https://stackoverflow.com/a/7557028" \t "_new)

## 4. Validación de fechas y tiempos con datetime

Marco teórico:  
datetime permite representar y operar con fechas y horas. Soporta comparación, conversión desde string y cálculo de diferencias temporales.

Aplicación en el proyecto:  
Se usó para determinar si un alquiler está activo, calcular vencimientos, validar rangos de reportes y emitir multas.

| **Objetivo del uso** | | **Herramienta utilizada** | **Prompt o pregunta** | |
| --- | --- | --- | --- | --- |
| Validar si una fecha es anterior o posterior a la actual usando datetime. | | ChatGPT | ¿Cómo comparar fechas con datetime en Python y detectar vencimientos? | |
| **Respuesta** | **¿Cómo usó o adaptó la respuesta?** | | | **Reflexión crítica** |
| Usar datetime.strptime y datetime.now() para comparar. | Lo apliqué directamente para verificar si un alquiler está vencido. | | | La IA dio ejemplos funcionales y correctos. Aclaró formatos y me ayudó a evitar errores. |

**Fuentes:**

* Python Datetime: [https://docs.python.org/3/library/datetime.html](https://docs.python.org/3/library/datetime.html" \t "_new)

## 5. Envío de correos con smtplib

Marco teórico:  
smtplib permite establecer conexión con un servidor SMTP (como Gmail) para enviar mensajes de correo. Se combina con email.message para personalizar los mensajes y adjuntar archivos.

Aplicación en el proyecto:  
Se usa para enviar confirmaciones de alquiler, extensiones de tiempo y notificaciones de multas con PDF.

| **Objetivo del uso** | | **Herramienta utilizada** | **Prompt o pregunta** | |
| --- | --- | --- | --- | --- |
| Enviar correo con adjunto PDF usando Gmail SMTP. | | ChatGPT | ¿Cómo usar smtplib para enviar un correo con archivo adjunto en Python? | |
| **Respuesta** | **¿Cómo usó o adaptó la respuesta?** | | | **Reflexión crítica** |
| Explicó uso de EmailMessage y configuración SMTP. | Adapté el código para usarlo con archivos reportlab. | | | Muy útil y preciso. Solo ajusté los nombres de archivo y credenciales. |

**Fuentes:**

* Python Email Docs: [https://docs.python.org/3/library/email.html](https://docs.python.org/3/library/email.html" \t "_new)
* Gmail SMTP Setup: https://realpython.com/python-send-email/

## 6. Generación de PDF con reportlab

Marco teórico:  
reportlab permite generar archivos PDF desde Python con texto, tablas y gráficos. Usa coordenadas absolutas o estructuras de layout como Table para componer documentos.

Aplicación en el proyecto:  
Se utiliza para generar reportes PDF de ingresos, listas de espacios y multas emitidas.

| **Objetivo del uso** | **Herramienta utilizada** | | **Prompt o pregunta** | |
| --- | --- | --- | --- | --- |
| Generar PDF con reportlab en Python. | ChatGPT | | ¿Cómo creo un PDF con tabla y texto usando reportlab en Python? | |
| **Respuesta** | | **¿Cómo usó o adaptó la respuesta?** | | **Reflexión crítica** |
| Mostró ejemplo básico de canvas, drawString, y drawTable. | | Tomé los métodos y los integré en mis funciones de reporte. | | Fue un gran punto de partida. Luego personalicé estilos y márgenes. |

### Fuentes:

* ReportLab Docs: https://www.reportlab.com/docs/reportlab-userguide.pdf

## 7. Uso de ttk.Treeview

Marco teórico:  
Treeview es un widget de ttk (Themed Tk) para mostrar datos tabulares. Soporta columnas, títulos, scroll, y es ampliamente usado para mostrar listas de datos.

Aplicación en el proyecto:  
Se usa para mostrar lista de espacios, vehículos, reportes y filtros de parqueo.

### Fuentes:

* TkDocs Treeview: https://tkdocs.com/tutorial/tree.html

## 8. Cifrado de contraseñas con hashlib

Marco teórico:  
hashlib permite usar algoritmos como SHA-256 para generar hashes de cadenas. Es fundamental para evitar almacenar contraseñas en texto plano.

Aplicación en el proyecto:  
Las contraseñas se cifran con SHA-256 antes de guardarse. En el login, se compara el hash ingresado contra el guardado.

### Fuentes:

* Python Hashlib Docs: [https://docs.python.org/3/library/hashlib.html](https://docs.python.org/3/library/hashlib.html" \t "_new)

## 9. División en módulos reutilizables

Marco teórico:  
Dividir la lógica en módulos permite mayor organización, pruebas más sencillas, y evita duplicar código. Cada módulo puede ser importado en distintas interfaces.

Aplicación en el proyecto:  
Se crearon módulos como:

* modulo\_utiles.py: validaciones, lectura/escritura JSON.
* modulo\_usuarios.py: lógica CRUD de usuarios.
* modulo\_reportes.py: generación de PDFs compartidos.
* modulo\_multas.py: gestión de multas, correo y generación.

### Fuentes:

* PEP8 Python Modules: [https://peps.python.org/pep-0008/#imports](https://peps.python.org/pep-0008/" \l "imports" \t "_new)

## Conclusiones del trabajo

## Problemas encontrados y soluciones aplicadas

### Cambio de estructura del archivo pc\_espacios.json

* + **Problema:** La estructura inicial del archivo no contenía todos los campos requeridos por la especificación oficial del sistema (como fecha de inicio, tiempo, fecha de fin, etc.).
  + **Solución:** Se refactorizó la estructura del archivo para cumplir con los requerimientos. Se ajustaron todas las partes del sistema que dependían del archivo (usuarios, administradores, inspectores) y se creó una función de migración automatizada.

### Envío de correos en entornos sin conexión o sin acceso SMTP

* + **Problema:** Durante pruebas en algunas máquinas, no se pudo establecer conexión con servidores SMTP por políticas de red.
  + **Solución:** Se agregó manejo de excepciones y mensajes claros al usuario. Se permite continuar sin correo si el fallo no es crítico.

### Validación de alquileres vencidos

* + **Problema:** Fallos de formato en fechas y errores al comparar horas provocaban errores silenciosos en la revisión de espacios.
  + **Solución:** Se reforzó el uso de datetime.strptime y validaciones con mensajes explícitos. Además, se manejaron fechas vacías y formatos erróneos.

### Renderizado de múltiples marcos en Tkinter

* + **Problema:** Al cambiar de pantalla en algunas funciones (ej. Configuración), se apilaban marcos sin destruir el anterior.
  + **Solución:** Se centralizó el control de pantallas con un método cambiar\_frame() que destruye el marco activo antes de crear el nuevo.

### Adaptación de reportes a PDF

* + **Problema:** Integrar datos dinámicos en reportes reportlab requería manejar tablas con estructuras variables.
  + **Solución:** Se modularizó la lógica de reportes en modulo\_reportes.py, reutilizando funciones para generar encabezados, estilos y listas.

## Aprendizajes obtenidos

### Dominio de tkinter a nivel intermedio-avanzado

* + Desde el manejo básico de ventanas hasta la gestión dinámica de frames, control de eventos, y componentes como Treeview.

### Uso profesional de JSON como almacenamiento

* + Se aprendió a usar estructuras anidadas con validaciones robustas, y a diseñar un flujo de datos coherente en el tiempo.

### Implementación modular escalable

* + Dividir el proyecto en módulos (usuarios, multas, reportes, etc.) permitió mantener el código limpio, reutilizable y fácil de mantener.

### Integración de tecnologías externas

* + Generación de PDFs con reportlab, envío de correos con smtplib, cifrado de contraseñas con hashlib.

### Aplicación práctica del desarrollo iterativo

* + A lo largo del proyecto se trabajó en ciclos: prototipo → prueba → corrección → extensión. Esto permitió mayor estabilidad y eficiencia.

## Estadística de tiempos invertidos

| **Actividad** | **Horas invertidas** |
| --- | --- |
| Análisis del problema | 5 horas |
| Diseño de algoritmos | 6 horas |
| Investigación de tecnologías | 4 horas |
| Programación (backend(módulos) + GUI) | 22 horas |
| Documentación interna | 2 horas |
| Pruebas de funcionalidad | 6 horas |
| Elaboración del manual de usuario | 3 horas |
| Elaboración de documentación final | 3 horas |
| Revisión y ajustes finales | 4 horas |
| **Total general estimado** | **55 horas** |

## Lista de revision del Proyecto

| **Concepto Evaluado** | **Pts** | **Avance (%)** | **Puntos Obtenidos** | **Análisis de Resultados** |
| --- | --- | --- | --- | --- |
| Configuración del parqueo | 4 | 100 | 4 | Funcionalidad implementada correctamente, con opción de actualizar o cancelar los cambios. Se validan formatos y tipos de datos. |
| Espacios de parqueo | 6 | 100 | 6 | Se implementó gestión visual y edición de espacios en formato horizontal con validación de datos y guardado en JSON. |
| Reporte ingresos de dinero | 5 | 100 | 5 | Funcionalidad completa con filtros por rango de fecha, agrupación por día y total acumulado. Exportable en texto y PDF. |
| Lista de espacios - todos | 1 | 100 | 1 | Implementada en reportes de usuario, inspector y administrador. |
| Lista de espacios - ocupados | 2 | 100 | 2 | Filtrado correcto según fecha-hora actual. Muestra placa, tiempos y espacio. |
| Lista de espacios - vacíos | 2 | 100 | 2 | Correcta clasificación y conteo. |
| Historial de espacios usados | 5 | 100 | 5 | Mostrado por día, descendente. Incluye duración y costos. Exportable a PDF. |
| Historial de multas | 4 | 100 | 4 | Se consulta por rango de fechas. Se lista detalle y monto. Incluye total final. |
| Iniciar sesión | 2 | 100 | 2 | Inicio de sesión por correo, validando contraseña cifrada. |
| Restablecer contraseña | 3 | 100 | 3 | A través del correo registrado. |
| Registrarse (CRUD usuarios) | 9 | 100 | 9 | Incluye creación, modificación, eliminación y agregado de vehículos. |
| Cifrado de contraseña | 4 | 100 | 4 | Implementado con hashlib.sha256. |
| Actualización de datos | 2 | 100 | 2 | Desde la opción perfil del usuario. |
| Envío de correo | 1 | 100 | 1 | Envío al alquilar, agregar tiempo, y generar multa (si aplica). |
| Parquear | 9 | 100 | 9 | Verifica disponibilidad, muestra alerta si está ocupado, calcula tiempo, registra y actualiza diccionario y archivo. |
| Envío de correo al parquear | 1 | 100 | 1 | Contiene datos de alquiler. |
| Agregar tiempo | 5 | 100 | 5 | Funcional y validado, suma al tiempo restante. |
| Envío de correo al agregar tiempo | 1 | 100 | 1 | Se notifica al usuario con detalles actualizados. |
| Desaparcar | 5 | 100 | 5 | Borra los datos del diccionario y actualiza archivo. Se puede ejecutar incluso si ya venció el tiempo. |
| Buscar parqueos disponibles | 1 | 100 | 1 | Filtra espacios habilitados y libres. |
| Historial de espacios usados (usuario) | 1 | 100 | 1 | Filtro por correo del usuario, se listan alquileres previos. |
| Historial de multas (usuario) | 1 | 100 | 1 | Muestra multas asociadas a placas del usuario. |
| Generar datos de la multa | 4 | 100 | 4 | Contiene motivo, fecha, espacio, placa. Validación de tiempo o placa. |
| Emitir el PDF | 1 | 100 | 1 | Implementado con reportlab, genera documento imprimible. |
| Enviar correo con la multa | 1 | 100 | 1 | Si la placa está asociada a un usuario registrado. |
| Lista de espacios de parqueos (inspector) | 1 | 100 | 1 | Compartida con administrador, implementada vía módulo. |
| Historial de multas (inspector) | 1 | 100 | 1 | Visualiza todas las multas por fecha, con detalle y total. |
| Creación y uso de módulos (modulo\_utiles, reportes) | 5 | 100 | 5 | Excelente modularización, todo reutilizable. |
| Validación de datos y procesos | 8 | 100 | 8 | Validaciones robustas de entradas, tipos, formatos, estados. |
| Documentación del proyecto | 5 | 100 | 5 | Se entregará documento en Word con todos los elementos solicitados. |

**Link al GitHub:**

[**https://github.com/Villarley/parqueos**](https://github.com/Villarley/parqueos)