

**Universidad Tecnológica de La Habana “José Antonio Echeverría”
Facultad de Ingeniería Informática**



“APLICACIÓN WEB PARA LA GESTIÓN DE HOSPEDAJES EN CUBA”

Trabajo de diploma para optar por el título de Ingeniería Informática

Autor: Héctor Alonso Remedios

hector@infocap.cu

Tutor: Ing. Darián Díaz García

Empresa de Telecomunicaciones de Cuba (ETECSA)

darian.diaz@etecsa.cu


La Habana, Cuba

Junio, 2019

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a "Casa Lourdes" y a la Facultad de Ingeniería Informática para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los 7 días del mes de junio del 2019.

Héctor Alonso Remedios 

Nombre completo del autor

Darian Díaz García 

Nombre completo del tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado **APLICACIÓN WEB Y ANDROIDE PARA LA GESTIÓN DE HOSPEDAJES EN CUBA**, fue realizado en nuestra entidad "Renta Lourdes". Se considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

☒ Totalmente

☐ Parcialmente en un _____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

- Se evita invertir en la compra de ambos softwares.
- Complemento de la estrategia de marketing, siendo un canal de fácil acceso, cómodo y de fácil uso.
- Potenciador en la entrada de ingresos. Al llegarle a más usuarios, la probabilidad de tener nuevos clientes que soliciten sus servicios es mayor.
- Centralización de la información
- Vía de comunicación directa. Los usuarios podrán efectuar votaciones, comentarios y la comunicación directa con el anfitrión.
- Ahorro en costos de publicidad.
- Adaptación al entorno móvil y web.

Como resultado de la implantación de este trabajo se reporta un efecto económico que asciende a 29941.00 CUP o 1197.00 CUC, al ser desarrollado por dos trabajadores del negocio, que como autores ofrecen gratuitamente la implementación y mantenimiento de ambos softwares.

Y para que así conste, se firma la presente a los 5 días del mes de junio del año 2019

Lourdes Villanueva García

Nombre del representante de la entidad

Propietaria arrendadora

Cargo

Logan 72052810617

Firma

Cuño

Agradecimientos:

Este trabajo representa la culminación de una importante etapa en mi vida, en la que han formado parte algunas personas. Ante todo quiero agradecer a Dios por darme fuerzas en el transcurso de la carrera. A mi esposa, por haberme incitado a continuar estudiando y por su apoyo incondicional. A mis padres, por formarme como la persona que soy, por mantenerse a mi lado durante todos estos años.

A mi familia en general por alentarme a seguir adelante. A mi compañero de estudios (José Manuel Cepero González), por su paciencia y sus excelentes consejos. A Darián Díaz, mi tutor, por enseñarme a hacer las cosas con excelencia, por escucharme, por cada uno de los consejos que me brindó. A todos mis compañeros de estudio, por compartir cada vivencia y guardar buenos recuerdos de estos años. A todos los profesores que influyeron en mi formación como ingeniero. A mis compañeros de trabajo, por ayudarme en cuanto requiera y brindarme todo el tiempo que necesitaba para estudiar. A todos, gracias...

Dedicatoria:

A Dios primeramente; a mi esposa e hija; a mis padres, mi hermano y mi tío. Por darme la vida, todo su amor y forjarme tal como soy.

Resumen

Un hospedaje es una morada en la que se renta una o varias habitaciones, espacios o la vivienda en su totalidad. Existen dos modalidades de alojamiento en Cuba; una en la que se puede arrendar en pesos cubanos (en lo adelante CUP), y otra en pesos cubanos convertibles (en lo adelante CUC). El arrendador que opte por la modalidad de CUC, puede arrendar a personas residentes o no en Cuba; el que lo realice en la modalidad de CUP solo está autorizado a arrendar a cubanos y extranjeros residentes permanentes en Cuba. Estos sitios no solamente se emplean para que las parejas tengan privacidad, sino también para los que viajan a otras provincias y no tienen donde pasar una o dos noches, o los que tienen que resolver determinado trámite y necesitan una habitación para cambiarse de ropa, asearse u otras necesidades.

El presente trabajo está esencialmente orientada al desarrollo de una aplicación web que contenga un registro de los hospedajes en Cuba. Entre las opciones que brinda el software se encuentran: Reservas a través de diferentes vías, llamada, mensaje y correo; disponibilidad de un mapa vectorial para ubicar geográficamente las viviendas; inscripción gratuita de los propietarios de las mismas sin límites geográficos nacionales; sincronización y análisis de datos; vínculo con una aplicación para teléfonos inteligentes; entre otras.

El software tiene entre sus metas: unificar la vía de promoción extendiéndola a nivel nacional y satisfacer tanto a los usuarios que necesitan un alojamiento como a los que rentan; logrando además incremento de ingresos en el sector privado.

Palabras claves: Aplicación Web, Hospedaje, Arrendador, Mapa, Trabajo por cuenta propia, CUP, CUC.

Summary

A lodging is a space in which one or several rooms, spaces or the whole house is rented. There are two types of accommodation in Cuba; one in which it can be leased in Cuban pesos (hereinafter CUP), and another in Cuban convertible pesos (hereinafter CUC). The lessor, who opts for the CUC modality, can lease residents or not in Cuba; whoever does it in the CUP modality is only authorized to rent to Cubans and foreigners who are permanent residents in Cuba. These sites are not only used for couples to have privacy, but also for those who travel to other provinces and have no place to spend one or two nights, or those who have to solve a certain procedure and need a room to change their clothes, wash themselves or other needs.

The present work is essentially oriented to the development of a web application that contains a vast registry of the lodgings in Cuba. Among the options offered by the software are: Reservations through different channels, call, message and mail; availability of a vector map to geographically locate the houses; free registration of the owners of the same without national geographic limits; synchronization and data analysis; link with an application for smartphones; among other.

The software has among its goals: to unify the promotion route extending it nationally and to satisfy both the users who need a lodging and those who rent; also achieving income increases in the private sector.

,

Índice

INTRODUCCIÓN.....	14
CAPÍTULO 1 FUNDAMENTOS TEÓRICOS.....	19
1.1 INTRODUCCIÓN.	19
1.2 OBJETIVOS ESTRATÉGICOS DE LOS HOSPEDAJES PARTICULARES EN CUBA.	19
1.3 DESCRIPCIÓN DE LOS PROCESOS QUE SE EJECUTAN EN EL CAMPO DE ACCIÓN.	19
1.4 ANÁLISIS CRÍTICO DE LA EJECUCIÓN ACTUAL DE LOS PROCESOS.....	20
1.5 PROCESOS OBJETO DE AUTOMATIZACIÓN.....	20
1.6 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN.	21
1.7 TENDENCIAS Y TECNOLOGÍAS ACTUALES.....	26
1.7.1 Lenguaje unificado de modelado	26
1.7.2 Procesos de desarrollo de software	26
1.7.3 Herramienta CASE	27
1.7.4 Tipo de aplicación.....	28
1.7.5 Lenguaje de programación	29
1.7.6 Gestor de base de datos	31
1.7.7 Marco de desarrollo (framework).....	33
1.8 ANÁLISIS CRÍTICO DE LAS FUENTES Y BIBLIOGRAFÍAS UTILIZADAS (ESTADO DEL ARTE).	34
1.9 CONCLUSIONES.....	35
CAPÍTULO 2 MODELO DEL DOMINIO.....	36
2.1 INTRODUCCIÓN.	36
2.2 DEFINICIÓN DE LAS ENTIDADES Y LOS CONCEPTOS PRINCIPALES.	36
2.3 REGLAS DEL NEGOCIO A CONSIDERAR.	38
2.4 REPRESENTACIÓN DEL MODELO DEL DOMINIO.	40
2.5 CONCLUSIONES.....	41
CAPÍTULO 3 REQUISITOS.	42
3.1 INTRODUCCIÓN.	42
3.2 DEFINICIÓN DE LOS REQUISITOS FUNCIONALES Y DE SEGURIDAD.....	42
3.3 ACTORES DEL SISTEMA A AUTOMATIZAR.	45
3.4 JERARQUÍA DE ACTORES.	45
3.5 PAQUETES Y SUS RELACIONES.....	46
3.6 DIAGRAMA DE CASOS DE USO DEL SISTEMA A AUTOMATIZAR.....	47
3.7 DESCRIPCIÓN DE LOS CASOS DE USO.	50
3.7.1 Paquete Seguridad	50

3.7.2	<i>Paquete Nomencladores</i>	51
3.7.3	<i>Paquete Gestión</i>	51
3.7.4	<i>Paquete Reportes</i>	53
3.7.5	<i>Paquete Rest.</i>	53
3.8	DEFINICIÓN DE LOS REQUISITOS NO FUNCIONALES.	54
3.9	CONCLUSIONES.	55
CAPÍTULO 4 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA		57
4.1	INTRODUCCIÓN.	57
4.2	DIAGRAMA DE CLASES DEL DISEÑO.	58
4.2.1	<i>Paquete Seguridad</i>	58
4.2.2	<i>Paquete Nomencladores</i>	59
4.2.3	<i>Paquete Reportes</i>	60
4.2.4	<i>Paquete Gestión</i>	61
4.2.5	<i>Paquete Rest.</i>	63
4.3	DISEÑO DE LA BASE DE DATOS.	64
4.3.1	<i>Modelo lógico de datos.</i>	64
4.3.2	<i>Modelo físico de datos.</i>	65
4.4	PRINCIPIOS DE DISEÑO.	66
4.4.1	<i>Interfaz de usuario.</i>	66
4.4.2	<i>Formato de salida de los reportes.</i>	67
4.4.3	<i>Ayuda.</i>	68
4.5	PATRONES DE DISEÑO	69
4.6	TRATAMIENTO DE ERRORES.	72
4.7	DIAGRAMA DE DESPLIEGUE.	74
4.8	ESTRUCTURACIÓN EN CAPAS.	74
4.9	CONCLUSIONES.	75
CAPÍTULO 5 VALIDACIÓN Y FACTIBILIDAD DE LA SOLUCIÓN PROPUESTA		76
5.1	INTRODUCCIÓN AL CAPÍTULO.	76
5.2	PRUEBAS AUTOMATIZADAS AL SISTEMA.	76
5.3	ANÁLISIS ESTÁTICO DEL CÓDIGO	78
5.4	ANÁLISIS DE COSTOS Y BENEFICIOS.	79
5.5	BENEFICIOS TANGIBLES E INTANGIBLES	80
5.6	CONCLUSIONES.	80
CONCLUSIONES		81
RECOMENDACIONES		82

REFERENCIAS BIBLIOGRÁFICAS.....	83
BIBLIOGRAFÍA	85
GLOSARIO DE SIGLAS Y TÉRMINOS	88
ANEXO 1: DESCRIPCIÓN DE CASOS DE USO DEL SISTEMA.	I
ANEXO 2: PRUEBAS AUTOMÁTICAS A LA APLICACIÓN.....	V
ANEXO 3: ANÁLISIS ESTÁTICO AL CÓDIGO DEL FICHERO <i>VIEWS.PY</i>.	VIII

Índice de tablas

Tabla 1 Comparación entre sistemas similares.....	24
Tabla 2 Definición de actores del sistema a automatizar	45
Tabla 3 Descripción del caso de uso <Autenticación>.....	50
Tabla 4 Descripción del caso de uso <Administrar Usuario>	50
Tabla 5 Descripción del caso de uso <Registro de usuario>	51
Tabla 6 Descripción del caso de uso <Modalidad de Renta>.....	51
Tabla 7 Descripción del caso de uso <Realizar Comentario>.....	51
Tabla 8 Descripción del caso de uso <Realizar una Reservación>	52
Tabla 9 Descripción del caso de uso < Gestionar mensajes a suscriptores>	52
Tabla 10 Descripción del caso de uso < Exportar a PDF>	53
Tabla 11 Descripción del caso de uso < Generar Reporte>	53
Tabla 12 Descripción del caso de uso < Sincronización de la información>	53
Tabla 13 Esfuerzo del implementador del sistema.....	79
Tabla 14 Descripción del caso de uso <Cambiar Contraseña>	I
Tabla 15 Descripción del caso de uso <Salvar base de datos>	I
Tabla 16 Descripción del caso de uso <Restaurar base de datos>	I
Tabla 17 Descripción del caso de uso <Visualizar trazas>	II
Tabla 18 Descripción del caso de uso <Administrar roles>	II
Tabla 19 Descripción del caso de uso <Gestionar nomencladores>	II
Tabla 20 Descripción del caso de uso <Gestionar hospedaje>	III
Tabla 21 Descripción del caso de uso <Gestionar oferta>	III
Tabla 22 Descripción del caso de uso <Visualizar reportes>.....	III
Tabla 23 Descripción del caso de uso <Tasa de cambio>	IV

Índice de figuras

Ilustración 1 Logo RealCasaRenta	21
Ilustración 2 Logo casa-cuba.org	22
Ilustración 3 Logo hospedaje cubano	23
Ilustración 4 Logo rent.cu.....	24
Ilustración 5 Modelo de dominio	40
Ilustración 6 Jerarquía de Actores	45
Ilustración 7 Paquetes y sus relaciones.....	46
Ilustración 8 Diagrama de Caso de Uso del Paquete Configuración.....	47
Ilustración 9 Diagrama de Casos de Uso Paquete Nomencladores	48
Ilustración 10 Diagrama de Casos de Uso Paquete Gestión.....	49
Ilustración 11 Diagrama de Casos de Uso del Paquete Reportes	49
Ilustración 12 Diagrama de Casos de Uso del Paquete Rest	50
Ilustración 13 Diagrama de clases del paquete Seguridad	58
Ilustración 14 Diagrama de clases del paquete Nomencladores.....	59
Ilustración 15 Diagrama de clases del paquete Reportes	60
Ilustración 16 Diagrama de clases del paquete Gestión (primera parte)	61
Ilustración 17 Diagrama de clases del paquete Gestión (segunda parte).....	62
Ilustración 18 Diagrama de clases del paquete <i>Rest</i>	63
Ilustración 19 Modelo lógico de datos	64
Ilustración 20 Modelo físico de datos	65
Ilustración 21 Interfaz de usuario (front).....	66
Ilustración 22 Interfaz de usuario (administración).....	67
Ilustración 23 Reporte en formato PDF.....	68
Ilustración 24 Ayuda del sistema (front)	69
Ilustración 25 Función que crea el objeto.....	70
Ilustración 26 Clase que hereda de ModelForm.....	70
Ilustración 27 Patrón Experto reflejado en la clase ReferenceZone.....	71
Ilustración 28 Patrón Decorator reflejado en método referencezone_list	71
Ilustración 29 Error en los campos requeridos	72
Ilustración 30 Error de URL.....	73
Ilustración 31 Error validación de datos desde el servidor.....	73
Ilustración 32 Diagrama de despliegue	74

Ilustración 33 Estructuración por capas.....	74
Ilustración 34 Pruebas automatizadas, código en el archivo test.py.....	77
Ilustración 35 Resultado de las pruebas automatizadas.....	77
Ilustración 36 Resultado del análisis estático del código	78

Introducción.

Antecedentes

Con la incorporación del trabajo por cuenta propia en el modelo económico del país, se habilitaron modalidades que, cumpliendo con las regulaciones y normas tributarias brindan a la sociedad cubana un impulso importante en la mejora de la economía. Una de estas modalidades es la de Arrendamiento de Vivienda, Habitaciones y Espacios[1] a partir del modificativo al artículo 74 de la ley No. 65 “Ley General de la Vivienda” el que se cita a continuación:

Decreto-Ley No. 353 Modificativo De La Ley No. 65 “Ley General De La Vivienda”, de 23 de diciembre de 1988

Artículo Único. Modificar el Artículo 74 de la “Ley General de la Vivienda”, el que queda redactado de la manera siguiente: Artículo 74.1. Los propietarios pueden arrendar sus viviendas, habitaciones y espacios, a personas naturales y jurídicas, al amparo de lo establecido en la legislación civil común, siempre que esté en correspondencia con las regulaciones urbanas y territoriales vigentes, mediante precio libremente concertado y previa autorización de la Dirección Municipal de Trabajo.

Actualmente la cifra de arrendadores de vivienda o habitaciones en La Habana es 14603[2] en CUP y 10566[2] en CUC. Hay municipios que cuentan con un gran número de hospedajes, como son Boyeros, Arroyo Naranjo y La Habana Vieja. Si esta cifra se complementa con la del resto de las provincias del país sería un número considerablemente alto.

Para reservar un hospedaje primeramente se debe conocer dónde está ubicado (o sea, su dirección particular), luego dirigirse personalmente ante el propietario o arrendatario, por último solicitar una habitación por un plazo de tiempo determinado.

Algunas de estas viviendas prestan servicios adicionales como alquiler de piscina, gastronómicos, etc.

Situación Problemática.

Mientras que algunos de los hospedajes prefieren el anonimato, la mayoría necesita promoción, pues les resulta difícil aumentar los clientes. Muchos solo se conocen localmente, lo que impide el crecimiento del negocio y, por ende, los ingresos se mantienen estables y en ocasiones bajos. Por otra parte, muchos clientes por falta de conocimiento e información se ven obligados a dirigirse a un mismo hospedaje, lo que puede provocar gastos adicionales e inconformidades. En otras ocasiones se encuentran en lugares donde no conocen ningún hospedaje y tienen que retirarse o buscar soluciones no deseadas. Actualmente existen sitios web que promocionan los hostales en Cuba en CUC, desatendiendo los que alquilan en CUP.

Problema.

¿Cómo ofrecer información sobre los alojamientos de forma centralizada, para los usuarios y arrendadores a nivel nacional?

Objeto de estudio.

Se define como objeto de estudio en el escenario cliente/usuario, las leyes y normas que avalan el arrendamiento de viviendas con fines de alojamiento y en el escenario de la informática se centra en el estudio de la programación de aplicaciones web, así como las tecnologías y herramientas necesarias para el desarrollo de las mismas.

Campo de acción.Escenario cliente/usuario:

Procesos de inscripción de las casas de hospedaje. Análisis de la información sobre la actividad de hospedaje de interés a la sociedad, así como los atributos comparativos que mejoren la decisión final del cliente.

Escenario Informático:

Incluye lenguaje de programación y marcos de trabajo (*frameworks*) para el desarrollo de aplicaciones web, gestores de bases de datos, herramientas y lenguajes de modelado basados en la metodología del Proceso Unificado de Desarrollo de Software.

Objetivo general del trabajo.

Desarrollar una Aplicación Web que gestione y centralice la información de hospedajes de casas particulares en Cuba.

Objetivos específicos y tareas desarrolladas para cumplir cada uno de ellos.

Para dar cumplimiento al objetivo general se exponen los siguientes objetivos específicos, con sus correspondientes Tareas de Investigación:

1. Realizar un **análisis de la bibliografía** acerca del campo de acción y el objeto de estudio para establecer el marco teórico conceptual del trabajo.
 - a) Análisis minucioso de la bibliografía para fundamentar los procesos objeto de estudio.
 - b) Investigación de trabajos anteriores similares al propuesto.
 - c) Valoración de las herramientas informáticas a emplear.
2. **Modelar** el proceso de un Hospedaje.
 - a) Identificar las necesidades de los arrendatarios en cuanto a la promoción y la información que desean publicar.
 - b) Identificar las necesidades de los usuarios en cuanto a la información que desean tener a su disposición.
 - c) Identificar las reglas del negocio.
 - d) Elaborar el diagrama de dominio.
3. **Analizar y diseñar** la propuesta de solución.

- a) Realizar el levantamiento de los requisitos funcionales y no funcionales de la aplicación.
- b) Identificar los casos de uso y diseñar el prototipo del sistema.
- c) Identificación de los paquetes y sus relaciones.

4. Implementar la Aplicación Web para la gestión de Hospedajes.

- a) Definir los principios y patrones de diseño que garanticen el cumplimiento de los requisitos no funcionales.
- b) Realizar los diagramas de clases de diseño.
- c) Diseñar la estructura de la base de datos.
- d) Definir el diagrama de despliegue.
- e) Implementar las funcionalidades diseñadas.

5. Validar el sistema propuesto.

- a) Identificar los casos de pruebas y hacer el diseño de estas.
- b) Realizar las pruebas a la aplicación, así como el análisis estático del código base.
- c) Identificar los beneficios tangibles e intangibles que brinda la aplicación tanto a los arrendadores como a los clientes.
- d) Estimar el tiempo, costo y esfuerzo a tener en cuenta en la realización de la aplicación.

Valor práctico.

En el proceso de informatización de la sociedad, ocupa un gran valor el desarrollo de las aplicaciones que automaticen, gestionen, centralicen un negocio como tal.

Con el desarrollo de la aplicación, los dueños de los hospedajes pueden registrarse al sitio brindando sus potencialidades en el servicio de hospedaje nacional e internacionalmente; tienen la posibilidad de incorporar nuevos clientes conservando los usuales; se fortalecen y tienen conocimiento de la satisfacción generada y de la popularidad de su negocio. Los clientes que usen la aplicación

web tienen un amplio registro de hospedajes con su ubicación geográfica que incluye todas las provincias de Cuba, podrán escribir comentarios sobre el servicio recibido, así como iniciar el proceso de reserva desde cualquier punto WIFI con internet, valorar la casa en varios aspectos, donde su criterio no queda subvalorado, sino que incide en el análisis estadístico que se ofrece de las casas, información que permite establecer comparaciones y decidir cuál es la mejor opción.

Estructuración del contenido del documento con una breve explicación de sus partes.

Capítulo 1: Fundamentación Teórica. Presenta sistemas similares vinculados al campo de acción. Muestra el estudio de tendencias y tecnologías actuales y el análisis de la bibliografía consultada como fuente de información.

Capítulo 2: Modelación del dominio. Hace referencia a las reglas a considerar.

Capítulo 3: Requisitos. Aborda la descripción de los actores y casos de usos del sistema. Se definen los requisitos funcionales y no funcionales. Presenta los diagramas de casos de usos del sistema y se muestran los paquetes y sus relaciones.

Capítulo 4: Descripción de la solución propuesta. Presenta los principios de diseño a tener en cuenta en el desarrollo del sistema como: la interfaz de usuario, los formatos de salidas de los reportes y el tratamiento de errores. Muestra los diagramas de clases del diseño, el modelo lógico y físico de la base de datos y el modelo de despliegue.

Capítulo 5: Estudio de Factibilidad. Plasma el análisis de la factibilidad del sistema, a partir del tiempo real empleado y el costo del proyecto. Se exponen las pruebas automatizadas realizadas al sistema con sus resultados.

Capítulo 1 Fundamentos teóricos.

1.1 Introducción.

En el presente capítulo se describe detalladamente el objeto de estudio, lo que permite obtener una mayor visión del proyecto. Se fundamentan las tendencias y tecnologías actuales para el desarrollo de aplicaciones web y se especifica la selección de aquellas que mejor se ajustan.

1.2 Objetivos estratégicos de los hospedajes particulares en Cuba.

- Incrementar y potenciar los ingresos.
- Mejorar la posición comercial del hospedaje.
- Reforzar las relaciones con los principales clientes.
- Optimizar los costos minimizando el impacto sobre la experiencia del cliente.
- Optimizar la rentabilidad.
- Estandarización de la calidad.
- Evolución en internet.

1.3 Descripción de los procesos que se ejecutan en el campo de acción.

Existen tres tipos de procesos, cuando un cliente elige un alojamiento basado en sus preferencias; cuando un arrendatario da promoción a su hospedaje y por ultimo; la reservación de un alojamiento.

El proceso de elección, por parte del cliente, sea nacional o internacional, comienza con un análisis particular de aspectos como presupuesto, facilidades que ofrece el hospedaje, ubicación, entre otras características; que visualiza en un sitio web o catálogo. Mientras que el proceso de promoción de un hospedaje se realiza a través de recomendaciones de propios clientes, de sitios web o agencias; estos dos últimos pueden cobrar impuestos.

Los sitios web le ofrecen al cliente información, en algunos casos estadísticas, de cuáles hospedajes pudieran alcanzar sus expectativas, brindando la posibilidad de escoger entre tantos uno específico para su satisfacción. A su vez, los arrendatarios perciben un mayor ingreso económico con una información actualizada en internet de su galería, ofertas, servicios etc.

El proceso de reserva procede una vez elegida la casa, se establece un acuerdo verbal entre ambas partes; en algunos casos se presentan proformas de contratos, pactando la fecha de inicio y fin de la renta.

1.4 Análisis crítico de la ejecución actual de los procesos.

Luego del análisis de los procesos anteriores se detectaron las siguientes deficiencias:

- El país está en proceso de informatización de la sociedad[3], aunque es conveniente para la aplicación, aun es deficiente el modo de acceso y los precios de conexión son altos para la población.
- Los sistemas existentes solo promueven los hospedajes de la modalidad en CUC y actualmente no existe un sitio web local (con dominio **cu**) que abarque las dos modalidades (CUC y CUP).
- Las aplicaciones existentes más utilizadas engloban todas las modalidades de los cuentapropistas propiciando una información generalizada, no detallada, perdiendo la exquisitez de la promoción.

1.5 Procesos objeto de automatización.

- Proceso de búsqueda de un hospedaje según criterios como localidad, facilidades, zona de referencia, etc.
- Proceso de inscripción a una base de datos centralizada de los hospedajes, teniendo en cuenta todos los atributos que la identifican, así como la latitud y longitud que la ubica en el mapa para su localización.
- Proceso de generación de estadísticas para los clientes; que por su importancia constituye uno de los principales aportes del sistema,

posibilitando al cliente poder decidirse a cuál hospedaje dirigirse, así como al arrendatario, qué debe hacer para mejorar su hospedaje.

- Proceso de notificación a usuarios; genera diferentes notificaciones al usuario dependiendo del rol que ocupa.
- Proceso de actualización de la APK.

1.6 Sistemas automatizados existentes vinculados al campo de acción.

Ámbito internacional



Ilustración 1 Logo RealCasaRenta

RealCasaRenta[4] es un grupo legalmente constituido, que gestiona el alojamiento en Cuba a través de una amplia red de casas particulares ubicadas a todo lo largo del país.

Entre sus principales características están:

- Internacionalización de la lengua.
- Enlaces a redes sociales de internet
- Reserva online.
- Información al turista como: Cambio de moneda, Comunicaciones, Trámites Aeropuerto y Aduana.
- Posee un blog para el usuario que visita el sitio (Desactualizado)
- Términos y condiciones donde muestra al usuario cómo usar su sitio.
- No posee forma de registro de un hospedaje nuevo.
- Diseño de interfaz de usuario llamativo y actual.
- En la vista en detalle de la casa posee un mapa de google y se pueden realizar comentarios.



Ilustración 2 Logo casa-cuba.org

Casas-cuba.org[5] es una agencia que te permite conocer Cuba y sus costumbres a través de una manera diferente a como generalmente los operadores del turismo la proponen, donde su principal objetivo es promover las casas particulares de renta.

Entre sus principales características están:

- Lista los hospedajes, sale por provincia.
- Internacionalización en el lenguaje.
- Enlace a *Facebook*.
- No tiene proceso de inscripción, por lo que debe ser por correo.
- Muestra otros servicios a parte de las rentas de las casas, como taxi en el aeropuerto, rentar un carro, taxi con chofer, paseo a caballo, etc.
- Se reserva online.
- Se pueden realizar comentarios sobre la casa.
- Posee informaciones de interés al turista como:
 - Antes de ir a Cuba - ¿Qué cosa es necesario saber?
 - Informaciones útiles para su viaje a Cuba.
 - ¿Cómo reservar las casas particulares en Cuba?
 - Zona de Alojamiento en la Habana.
 - Aduana - Informaciones acerca de la Aduana de Cuba.
 - Visa o Tarjeta de Turista - ¿Cómo se obtiene?
 - Playa Varadero, algunos la llaman la más bella del mundo.
 - ¿Cómo vestirse en Cuba?
 - En el aeropuerto desde el momento en que aterriza el avión.



Ilustración 3 Logo hospedaje cubano

HospedajeCubano.com[6] Renta de habitaciones en Cuba. Se puede encontrar variada información que necesita un turista para realizar su viaje a Cuba.

En sus páginas obtendrá ofertas de casas particulares para hospedarse en cualquier rincón de Cuba y los lugares de interés turístico más próximos a cada una de las casas que promocionan.

Algunas de sus características son:

- No posee una interfaz de usuario actualizada.
- Se reserva online.
- Información para el usuario como:
 - Consejos Útiles
 - Viajar a Cuba
 - Casa particular
 - Renta privada
 - Renta de Autos
 - Transporte en Cuba
 - Cambio de moneda
 - Clima en Cuba
- Otros servicios
 - Servicio de taxis
 - Foros de discusión
 - Opiniones
 - Encuestas
 - Sugiere un hostel
 - Recomiéndanos
 - Envía una postal
 - Noticias

- Intercambia enlaces
- Sitios amigos
- Directorios y enlaces
- Enlaces relacionados
- No tiene forma de registrarse como arrendatario o usuario y crear un hospedaje nuevo.

Ámbito nacional:



Ilustración 4 Logo rent.cu

Rent.cu[7] & *Havanacity* es un equipo de trabajo con representantes en los principales polos de Cuba, con el objetivo de que sus clientes y visitantes encuentren la información que necesitan para su viaje a Cuba.

Entre sus principales características están:

- Tiene suscripción al sitio.
- Lista las casas de renta por provincia.
- Internacionalización de la lengua.
- Enlace a *Facebook*.
- El proceso de inscripción no se encuentra completo en el sitio, solo pide nombre, correo y mensaje; dando lugar a muchos pasos detrás para inscribirse.

Tabla 1 Comparación entre sistemas similares

Requisitos/App	RealCasaRenta	Casas-cuba	HospedajeCubano	Rent
Hospedaje con dominio cubano(.cu)	NO	NO	NO	SI

Registro de las casas de hospedajes	NO	NO	NO	NO
Visualización de la casa en un mapa	SI	SI	NO	NO
Diseño gráfico moderno y <i>responsive</i>	SI	SI	NO	SI
Interacción con una aplicación para teléfonos inteligentes	NO	NO	NO	NO
Realizar una reserva del hospedaje online	SI	SI	SI	SI
Visualización por parte del arrendatario de gráficos estadísticos referente a su hospedaje	NO	NO	NO	NO
% Cumplimiento de los Requisitos	42.9%	42.9%	14.3%	42.9%

La tabla anterior muestra algunos de los requisitos fundamentales a cumplir en el desarrollo de este proyecto y se puede apreciar que el cumplimiento por los sistemas automatizados existentes no sobrepasa en ninguno de los casos el 43%. La realización de una aplicación que permita el cumplimiento de estos

requisitos en su totalidad, ofrecerá al cliente o al arrendatario una mejor oferta para el negocio en cuestión.

1.7 Tendencias y tecnologías actuales.

Antes de comenzar con un producto, es indispensable elegir la metodología y procesos de desarrollo de software a seguir y el lenguaje de modelado, con el fin de garantizar la calidad, además de un sistema robusto y de fácil mantenimiento. A continuación, se analizan las tendencias en tecnologías de ingeniería de software, lenguajes de programación y marcos de desarrollo.

1.7.1 Lenguaje unificado de modelado

Lenguaje Unificado de Modelado (UML), es un lenguaje gráfico para visualizar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como son: procesos del negocio y funciones de sistemas, además de esquemas de base de datos y componentes de software reutilizables. [8]

1.7.2 Procesos de desarrollo de software

Proceso ágil de desarrollo de Software (**SCRUM**): es un proceso en el que se aplican, de manera regular, un conjunto de buenas prácticas para trabajar colaborativamente (en equipo) y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En SCRUM se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto.[9]

Proceso Unificado de desarrollo de Software (**RUP**)[10]: Es uno de los más utilizados para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso dirigido por casos de uso, avanza a través de una serie de flujos de trabajo que parten de ellos; está centrado en la

arquitectura y es iterativo e incremental. Se divide en 4 fases de desarrollo del software: inicio, elaboración, construcción y transición.

Ventajas[11]:

- Está basado totalmente en mejoras prácticas de la metodología.
- Reduce riesgos del proyecto.
- Incorpora fielmente el objetivo de calidad.
- Integra desarrollo con mantenimiento.

Desventajas[11]:

- Pretende prever y tener todo el control de antemano.
- El modelo genera trabajo adicional.
- Genera muchos costos.
- No recomendable para proyectos pequeños.

Selección:

Se decide utilizar RUP y como lenguaje de modelado a UML; puesto que es aplicable a proyectos de cualquier tamaño, siendo orientado al proceso, promueve que la arquitectura se defina tempranamente en el proyecto y hace especial énfasis en la definición del proceso: roles, actividades y artefactos. Además, este proceso de modelado fue impartido durante la carrera y se cuenta con los conocimientos necesarios para su adecuada utilización.

1.7.3 Herramienta CASE

Enterprise Architect (EA)[12]

Es una herramienta CASE para el diseño y construcción de sistemas de software. Cubre todos los aspectos del ciclo de desarrollo, con herramientas que le proporcionan una infraestructura enormemente competitiva en torno a la modelación del negocio, diseño de software, ingeniería de sistemas, arquitectura corporativa, gestión de requerimientos y pruebas.

Visual Paradigm[13]

Es una herramienta multiplataforma que emplea el lenguaje de modelado gráfico UML. Es un instrumento profesional que soporta el ciclo de vida completo del

desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Dentro de sus características principales están:

- Permitir generar código a partir del modelo de clases del diseño.
- Diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.
- Soporta aplicaciones web.
- Los diagramas generados se pueden modificar directamente desde los documentos.

Selección:

La herramienta CASE seleccionada para la modelación de la solución de software es el sistema Visual Paradigm, debido a su potencialidad en el modelado del lenguaje UML utilizando RUP (Rational Unified Process, Proceso Unificado de desarrollo), además por las facilidades que provee en el mapeo relacional de objetos (ORM) a partir del diagrama físico de la base de datos.

1.7.4 Tipo de aplicación

Se propone realizar una aplicación web[14]

Una aplicación web es una aplicación a la que se accede a través de internet u otras redes similares como intranet, que no requiere instalación para los usuarios. Las aplicaciones web tienen mucho éxito, principalmente porque solo requieren un navegador web independientemente del sistema operativo y no se necesita instalar ningún software en los equipos de cada usuario que las utilizan. Puede ser ejecutada en múltiples plataformas diferentes.

Ventajas:

- Permite realizar actualizaciones o hacer cambios en el software, es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor, lo que implica que no hay que distribuirla entre las demás PC.

- Para acceder a la aplicación solo se necesita un navegador web y la conexión puede ser desde cualquier máquina que esté conectada al servidor.
- Se pueden usar desde cualquier sistema operativo porque solo es necesario tener un navegador.
- El control se encuentra centralizado. Los accesos, recursos y la integridad de los datos son controlados por el servidor, de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- No hay problemas de compatibilidad. Basta tener un navegador actualizado para poder utilizarlas.
- Actualizaciones inmediatas. Como el software es gestionado por el propio programador, cuando el usuario se conecta, está usando siempre la última versión que haya sido lanzada.

1.7.5 Lenguaje de programación

Para el desarrollo de aplicaciones Web en el ámbito de software libre se destacan las alternativas de uso de Java, Python, y PHP.

A continuación se presenta un esbozo de estos lenguajes:

Java[15]:

Es un lenguaje de Programación Orientado a Objeto (POO). Es un lenguaje bien estructurado, sin punteros y sin necesidad de tener que controlar la asignación de memoria a estructuras de datos u objetos. Para desarrollar código Java se utiliza el Kit de Desarrollo de aplicaciones Java (Java™ Development Kit, JDK por sus siglas en inglés), el mismo es gratis y público. Las características más importantes de Java que lo hacen distinto de los demás lenguajes son:

- Fácil de aprender y bien estructurado.
- Aprovecha las características de la mayoría de los lenguajes modernos, evitando sus inconvenientes.
- Tiene gran funcionalidad gracias a sus bibliotecas (clases). El manejo de la memoria lo gestiona el propio programa y no el programador.

- Es robusto. Las aplicaciones creadas con Java son susceptibles de contener pocos errores.

Python[16]:

Es un lenguaje de scripting, independiente de la plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. Contiene tipos de datos y funciones, incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero. Se puede desarrollar en diversas plataformas como Linux, Windows, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, entre otros. Python es gratuito, incluso para propósitos empresariales, cuenta con amplia comunidad en la red. Su simpleza, legibilidad y similitud con el idioma inglés lo convierten en un gran lenguaje ideal para principiantes.

PHP[17]:

El Pre-Procesador de Hipertexto (PHP) es un lenguaje de código abierto utilizado para el desarrollo web de contenido dinámico y uno de los primeros en incorporarse directamente en los archivos HTML de las páginas de Internet. Lo mejor de utilizar PHP es su facilidad para los principiantes, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. Cuenta con variedad de framework para su desarrollo. Goza de popularidad, existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código. PHP es eficiente, con escaso mantenimiento y un servidor gratuito, puede soportar sin problema millones de visitas diarias. Se integra bien con múltiples gestores de bases de datos.

Selección:

Se escoge Python por ser un lenguaje gratuito y multiplataforma. Además de su posibilidad de acceso a muchos tipos de bases de datos. También es importante destacar su capacidad de crear páginas dinámicas, así como la posibilidad de

separar el diseño del contenido de una web. Posee como ventaja la fácil interpretación de código que ofrece visualmente y obliga a programar de manera ordenada y bien estructurada.

1.7.6 Gestor de base de datos

Un Sistema de Bases de Datos es una serie de recursos para manejar grandes volúmenes de información, sin embargo no todos los sistemas que manejan información son bases de datos[18].

El gestor de base de datos es el encargado de manejar los datos, el cual define la estructura para almacenar los datos y la manipulación de los mismos.

SQLite[19] es un sistema completo de bases de datos que soporta múltiples tablas, índices y vistas. No necesita un proceso separado funcionando como servidor, pues lee y escribe directamente sobre archivos que se encuentran en el disco duro. El formato de la base de datos es multiplataforma e indistintamente se puede utilizar el mismo archivo en sistemas de 32 y 64 bits. La base de datos se almacena en un único fichero a diferencia de otros DBMS, que hacen uso de varios archivos. Emplea registros de tamaño variable de forma tal que se utiliza el espacio en disco que es realmente necesario en cada momento. El código fuente está pensado para que sea entendido y accesible por programadores promedio. Todas las funciones y estructuras están bien documentadas. Existe un programa independiente de nombre SQLite que puede ser utilizado para consultar y gestionar los ficheros de base de datos SQLite. También sirve como ejemplo para la escritura de aplicaciones utilizando la biblioteca SQLite.

PostgreSQL[20] es un SGBD relacional orientado a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de una forma desinteresada, altruista, libre y, en algunos casos apoyado por organizaciones

comerciales. Cuenta en su instalación con un conjunto de herramientas para el trabajo con la base de datos, lo que evita la instalación de herramientas extras. PostgreSQL funciona muy bien con 18 grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, entre otros; aparte de permitir la creación de tipos propios. Posee múltiples métodos de autenticación, acceso encriptado vía SSL y permite el trabajo sobre las plataformas Linux, UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64 bit.

MySQL[21] es un sistema de gestión de base de datos relacional, multiproceso y multiusuario, con más de seis millones de instalaciones. MySQL, desde enero de 2008, una subsidiaria de Sun Microsystems desarrolla como software libre en un esquema de licenciamiento dual. Sin embargo, MySQL con respecto a la seguridad ofrece un sistema de contraseñas y privilegios seguros mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor. No obstante, no hay una instalación definida para el trabajo con la BD, debido a que MySQL se utiliza la mayoría de las veces dentro de otras instalaciones, muchas de ellas orientadas al desarrollo de aplicaciones web.

Selección:

Se selecciona como gestor de base de datos a PostgreSQL, porque está desarrollado bajo código abierto y es gratis su distribución. Al ser multiplataforma puede ser usado en cualquier sistema operativo, facilitando su uso tanto en el desarrollo de la aplicación como en el despliegue de la misma. Cuando se trata de manejar bases de datos con alto volumen de información ofrece gran rendimiento[22]. Evita concurrencia gracias al método de Control de Concurrencias Multiversión (o por sus siglas en inglés MVCC), característica primordial para el software.

1.7.7 Marco de desarrollo (*framework*)

Django[23], [24] es un framework diseñado para hacer tareas comunes de desarrollo web de forma rápida y fácil. Aunque es posible utilizarlo sin una base de datos, soporta conexión con múltiples sistemas de base de datos. Cuenta con un ORM (Mapeo-Objeto-Relacional), en el que es posible definir la estructura de la base de datos utilizando código Python, para lo que define los modelos y una vez definidos, permite crear automáticamente una interfaz administrativa, que le brinda a los usuarios la posibilidad de añadir, modificar y eliminar objetos. Posee una ruta de búsqueda de plantillas, lo que le permite al desarrollador minimizar redundancia entre las plantillas y se basa en patrón de arquitectura MVC (Modelo-Vista-Controlador).

TurboGears[25] es un framework para desarrollo web de código abierto, escrito en Python. Es un stack web completo, que abarca desde el Javascript del cliente hasta un mapper relacional-objetos para la base de datos. Crear una aplicación lista para extender con bases de datos en cuestión de minutos. Su última versión 1.0 incluye templates (además del oficial se puede usar otros vía plug-ins), AJAX, servidor (mappeador) web, manejo de formularios, widgets (con navegador incluido), interfaz para diseñar y administrar la base de datos, con interfaz web para hacer las traducciones (y hasta para comenzar con una traducción automática de google), autenticación/permisos, creación simple de interfaz administrativa (ABM simple), etc.

Flask[26], [27] es un microframework que se esfuerza por ser simple y pequeño; todo el framework consiste en un puñado de módulos. No hay un esqueleto o una estructura de la cual partir, todo se empieza con una página en blanco. Flask no proporciona grandes funcionalidades, pero hay extensiones Flask disponibles para agregar ORM, validación de formularios, manejo de carga, etc. Flask es ideal para:

- Aprender a programar.

- Los desarrolladores que se preocupan por las buenas prácticas y el código "elegante".
- Los desarrolladores que quieran crear prototipos de forma rápida.
- Los desarrolladores que necesitan una aplicación independiente. [20]

Selección:

Se selecciona Django, pues es un framework diseñado para hacer tareas comunes de desarrollo web de forma rápida y fácil. Posee una ruta de búsqueda de plantillas, lo que le permite al desarrollador minimizar redundancia entre las plantillas. Se basa en el patrón de diseño MVC (Modelo-Vista-Controlador), pero realmente es el modelo MTV (Modelo-Plantilla-Vista) y posee una biblioteca *Django-RestFramework* que facilita la intercomunicación con la API, aspecto fundamental para llevar a cabo el desarrollo del software.

1.8 Análisis crítico de las fuentes y bibliografías utilizadas (Estado del arte).

Para la conformación del presente trabajo se han consultado diversas fuentes bibliográficas, entre las que se encuentran: libros, tesis de pregrado y artículos publicados en sitios confiables de internet relacionados con el tema. La bibliografía consultada cuenta con reconocidos profesores y especialistas, que son referenciados en numerosas publicaciones de diversos tipos, dando muestra de la confiabilidad de sus criterios. En la actualidad existe además de una alta disponibilidad de tecnologías a emplear en el desarrollo de productos de software, una amplia variedad de criterios, razón por la que se hace compleja la elección de una específica. Después de un arduo trabajo investigativo y del estudio de numerosas fuentes bibliográficas, se ha podido determinar los resultados de la puesta en práctica. Para el desarrollo de la aplicación, se demostró el uso de la metodología y las herramientas empleadas, logrando resultados con impacto positivo en el cumplimiento de los requisitos del sistema propuesto.

1.9 Conclusiones.

- a) Con el análisis exhaustivo de la bibliografía consultada, se determinan para el desarrollo de la aplicación las leyes y normas que avalan el arrendamiento de viviendas con fines de alojamiento, así como las tecnologías y herramientas necesarias.
- b) Se investigaron cuatro sistemas similares al propuesto; obteniendo una tabla comparativa, en la que se evidencia que no existen sistemas informáticos que ofrezcan una solución completa a las necesidades presentadas.
- c) Fueron seleccionadas las siguientes tecnologías: UML, RUP, Visual Paradigm, Python, Django y PostgreSQL para el óptimo desarrollo de la Aplicación Web, argumentando cada selección.

Por lo antes expresado, se propone desarrollar una aplicación que ofrezca de manera integral mayores beneficios a la gestión del proceso de hospedaje en Cuba.

Capítulo 2 Modelo del dominio.

2.1 Introducción.

En el presente capítulo se desarrolla un modelo de dominio donde se presentan los conceptos importantes y las entidades del sistema.

El modelo de dominio puede ser tomado como el punto de partida para el diseño de un sistema basado en la programación orientada a objetos. El funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo de dominio constituye una primera versión del sistema. Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis, como paso previo al diseño de un sistema. El modelo de dominio es utilizado por el analista como un medio para comprender el sector de negocios al que el sistema va a servir.

Si no es aplicable el modelo del negocio, entonces puede procederse a identificar conceptos, se les da definiciones a estos conceptos y se trata de unir o relacionar en otro modelo distinto que es el de dominio. Este modelo permitirá mostrar de manera visual los principales conceptos que se manejan, ayudando a los usuarios, desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se desarrolla el sistema.

2.2 Definición de las entidades y los conceptos principales.

El modelo de dominio permite comprender el contexto del sistema a través de la descripción de conceptos importantes, a través de objetos del dominio, y del enlace entre estos. Estos objetos pueden ser clases conceptuales, así como eventos que suceden en el entorno del sistema. A continuación, se detallan los conceptos fundamentales que permiten comprender el contexto de este trabajo.

Provincia: Las provincias del país Cuba, sería un nomenclador de la aplicación. Una casa pertenece a un municipio y este último a una provincia.

Municipio: Todos los municipios del país Cuba, nomenclador, identificados por su provincia. Una casa pertenece a un municipio.

Facilidades: Refiere a todo servicio que puede ofrecer una casa de hospedaje, donde el atributo nombre nos brinda la información pertinente. Ejemplo: Mucama, Agua fría y caliente, aire acondicionado etc. Una casa tiene varias facilidades.

Tipo de moneda: Nombres de las monedas, en su versión abreviada, utilizadas en las casas de cambio en Cuba y que, a modo de facilidad al cliente, las casas aceptan como pago; ejemplo: USD, CAD, CUC, CUP, etc.

Punto de interés: Son localidades, regiones, negocios u otros que están cercanos al hospedaje y son de interés al cliente para la toma de decisiones.

Tipo de punto de interés: Son los tipos que identifican esas localidades, regiones, negocios u otros, que agrupan en conjunto. Ejemplo: restaurantes, museos, etc.

Modo de renta: Son los tres tipos de modalidades que un hospedaje puede ejecutar: por habitación, por días y por horas.

Zona de referencia: El cliente debe y quiere saber dónde se encuentra ubicado el hospedaje y este dato lo brinda la zona de referencia. Ejemplo: Centro Histórico, Playa, Campestre, etc.

Reservación: Un usuario puede realizar la reserva de un hospedaje sin necesidad de estar registrado en el sistema, simplemente llenando los datos requeridos, dicha reserva se le enviará al dueño del hospedaje vía correo.

Tasa de cambio: De cada tipo de moneda, cada día, qué valor representa la compra y venta. Dato útil cuando una casa renta acepta otros tipos de moneda al CUC.

Usuario: Entes registrados en el sistema.

Grupos: Roles de los usuarios.

Permisos: Los permisos que a cada usuario le es asignado en el sistema.

Oferta: Serían todas las ofertas que puede tener, en general, las casas de hospedaje. Dentro de la descripción de la misma se guarda todo lo referente a cada oferta en particular, ejemplo: "Usted puede rentar toda la noche por un precio de 10 cuc". Una casa puede o no tener oferta(s).

Galería: Se guarda todas las imágenes de los hospedajes.

Valoración (Rating): Un usuario registrado en el sistema puede dar su valoración del hospedaje visitado en un intervalo de 0 a 5 puntos.

Visitas: Registro de todas las visitas que tiene un hospedaje, validado que se registre en un día la visita de un usuario por el campo de identificación, que sería en este caso, el IP.

Favorita: Un usuario registrado puede definir uno o varios hospedajes como su favorito.

Comentarios: Son las opiniones almacenadas por los usuarios registrados en el sistema sobre el hospedaje. En caso de existir un comentario sin hospedaje, sería un comentario del sitio web.

Renta: Constituye el conjunto de hospedajes que se encuentran ubicados en Cuba.

2.3 Reglas del negocio a considerar.

- El administrador es el encargado de la seguridad en el sistema, dígase la gestión de los usuarios y roles, velar el estado del historial y realizar la salva diaria de la base de datos.
- Salva guarda de la base de datos del sistema por el administrador.
- El WebMaster asume la responsabilidad de gestionar los nomencladores necesarios para la inserción de los datos en el sistema, pues los mismos pertenecen al negocio como tal.
- El WebMaster debe activar o desactivar los hospedajes una vez que existan a modo de validación de datos, así también los comentarios realizados a los hospedajes y al sitio en general.
- El WebMaster puede insertar un hospedaje nuevo a pedido de cliente, como es necesario que exista el usuario registrado en el sistema, el actor tiene la facultad de crear en su sección un usuario con rol **arrendatario** (rol asignado automáticamente); el usuario será activado por el administrador.

- El web máster debe insertar diariamente la tasa de cambio manualmente desde el sitio www.cadeca.cu, pues no existe un **api** del que se pueda consumir estos datos.
- El sitio web tiene información dinámica, el WebMaster es el encargado de gestionar esos datos.
- Para realizar un comentario, se debe estar registrado en el sitio.
- Solo entrar al sitio de administración escribiendo la **url** específica, no puede tener un link desde el sitio.
- El usuario debe tener internet o red Cuba para navegar en el sitio.

2.4 Representación del modelo del dominio.

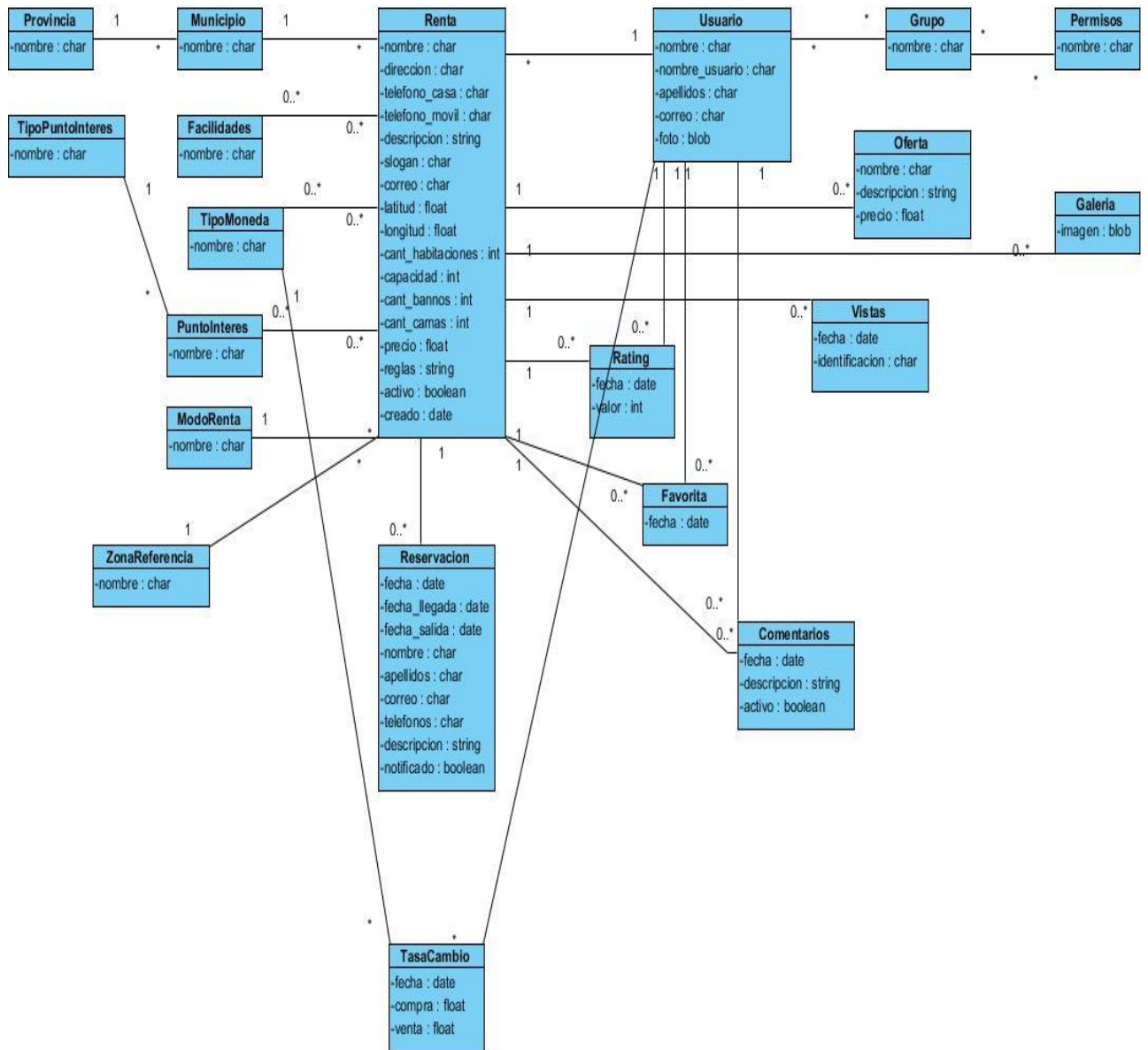


Ilustración 5 Modelo de dominio

2.5 Conclusiones.

- a) Se identificaron las necesidades de arrendadores en cuanto a la promoción e información que desean publicar como: imágenes, reglas o normas del hospedaje, facilidades que ofrecen, vía de comunicación (correo, teléfono), etc.
- b) Fueron identificadas las necesidades de los usuarios en cuanto a la información que desean tener a su disposición, como: imágenes, localización, vía de comunicación, capacidad del hospedaje, etc.
- c) A través de las reglas del negocio se determinó que, fundamentalmente, el usuario debe tener internet o red Cuba para navegar o administrar el sitio.
- d) Se confeccionó el diagrama de dominio, teniendo en cuenta las entidades que intervienen en el negocio (renta, reservación, usuario, oferta, facilidades, etc.).

Capítulo 3 Requisitos.

3.1 Introducción.

En el presente capítulo se definen todos los requerimientos con los que debe cumplir el software a desarrollar. También se incluye el diagrama, la descripción de los actores y de los casos de uso del sistema, así como los paquetes que lo componen y la relación que existe entre ellos.

3.2 Definición de los requisitos funcionales y de seguridad.

Paquete de Seguridad

1. Administrar perfil de Usuario.
 - a) Cambiar contraseña.
 - b) Modificar datos personales.
2. Administrar Roles.
3. Visualizar trazas.
4. Gestionar Permisos.
5. Salva guarda de la base de datos.

Paquete de Nomencladores

6. Nomencladores.
 - a) Listar provincias.
 - b) Listar municipios.
 - c) Gestionar punto de interés.
 - d) Gestionar tipo de punto de interés.
 - e) Gestionar facilidades.
 - f) Gestionar modalidad de renta.
 - g) Gestionar zona de referencia.
 - h) Gestionar tipo de moneda.

Paquete gestión de los hospedajes

7. Gestionar Casas de renta.

- a) Al insertar la casa, facilitar al usuario un mapa para la ubicación de su casa.
 - b) Cuando se muestra la casa mostrar un mapa con su ubicación.
- 8. Realizar el rating de las casas.
 - a) El cliente puede valorar una casa desde el sitio web.
- 9. Marcar una casa como favorita.
 - a) El cliente puede marcar una casa como su favorita.
- 10. Moderar comentarios.
 - a) Comentarios del sitio.
 - b) Comentarios de una casa en particular.
- 11. Gestionar suscripción del sitio.

Paquete de comunicación con la Apk

- 12. Recibir vía REST.
 - a) Los ratings de las casas realizados.
 - b) Las casas favoritas de casa usuario.
 - c) Comentarios realizados.
 - d) Cantidad de visitas de hospedaje.
- 13. Descargar la apk para teléfonos inteligentes.
- 14. Sincronizar base de datos ofreciendo servicio REST.
 - a) Obtener a partir de una fecha los cambios realizados en la base de datos.
 - b) Conformar el paquete a enviar para la sincronización.

Paquete de servicios del sitio

- 15. Gestionar informe a enviar al suscriptor.
- 16. Obtener las visitas diarias al sitio.
- 17. Cálculo del ranking de las casas con más comentarios.
- 18. Cálculo del ranking de las casas con mejor rating.
- 19. Cálculo del ranking de las casas favoritas.
- 20. Cálculo del ranking de las casas con más visitas.
- 21. Gestionar solicitud de reserva de una casa en particular.
 - a) El usuario puede llenar un formulario para reservar.

- b) Enviar solicitud de reserva por correo.

Paquete de notificaciones

22. Notificaciones

- a) Ingreso y cambios realizados en el perfil de usuario.
- b) Vía email para completar el registro de un usuario.
- c) Errores en el ingreso de los datos.
- d) Actualizaciones en las estadísticas de las casas.
 - I. Rating de la casa (Se mide con un valor de 5 puntos donde 5 es el máximo).
 - II. Comentarios nuevos de la casa.
- e) Nueva reserva por el cliente.

Paquete de reportes

- a) El listado ordenado por las casas con más comentarios.
- b) El listado ordenado por las casas con mejor rating.
- c) El listado ordenado por las casas favoritas.
- d) El listado ordenado por las casas más visitadas.

Paquete de gráficos

23. Gráficos.

- a) Visitas al sitio y visitas totales de los hospedajes vs. mes del año (Barra).
- b) Cantidad de valoraciones por hospedajes (Pie).
- c) Valoración por hospedajes ordenado descendientemente (Barra).
- d) Cantidad de comentarios por hospedaje ordenado descendientemente (Barra).
- e) Cantidad de visitas por hospedaje ordenado descendientemente (Barra).
- f) Cantidad de favoritos por hospedaje ordenado descendientemente (Barra).
- g) Entradas del sitio vs. entradas desde la apk (Pie).
- h) Usuarios activos vs. usuarios inactivos (Pie).

3.3 Actores del sistema a automatizar.

Tabla 2 Definición de actores del sistema a automatizar

Nombre	Descripción
Administrador	Es el responsable de administrar el sistema. Se encarga de la seguridad y el control de las acciones realizadas en el sistema a través de la gestión de usuarios, permisos, grupos y visualización de trazas.
WebMaster	Es el responsable de administrar el negocio. El encargado de activar o desactivar los hospedajes, así como los comentarios realizados del sitio. Gestiona la información dinámica, los nomencladores y el proceso de suscripción.
Arrendatario	Usuario con propiedad(es) de hospedaje. Gestiona su(s) hospedaje(s). Gestiona las ofertas que brinda la renta.
Usuario	Actor genérico que hace uso de funcionalidades que son comunes al de actores del sistema y por ende se crea para que hereden de él, generaliza el rol de autenticación al sistema, crea valoración de los hospedajes, así como comentarios. Puede crear un hospedaje.

3.4 Jerarquía de actores.

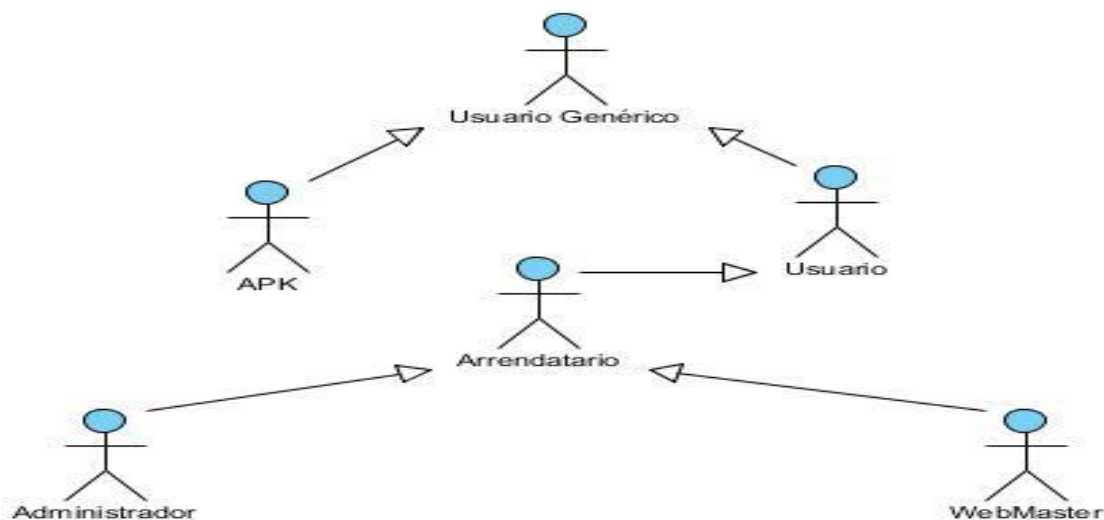


Ilustración 6 Jerarquía de Actores

3.5 Paquetes y sus relaciones.

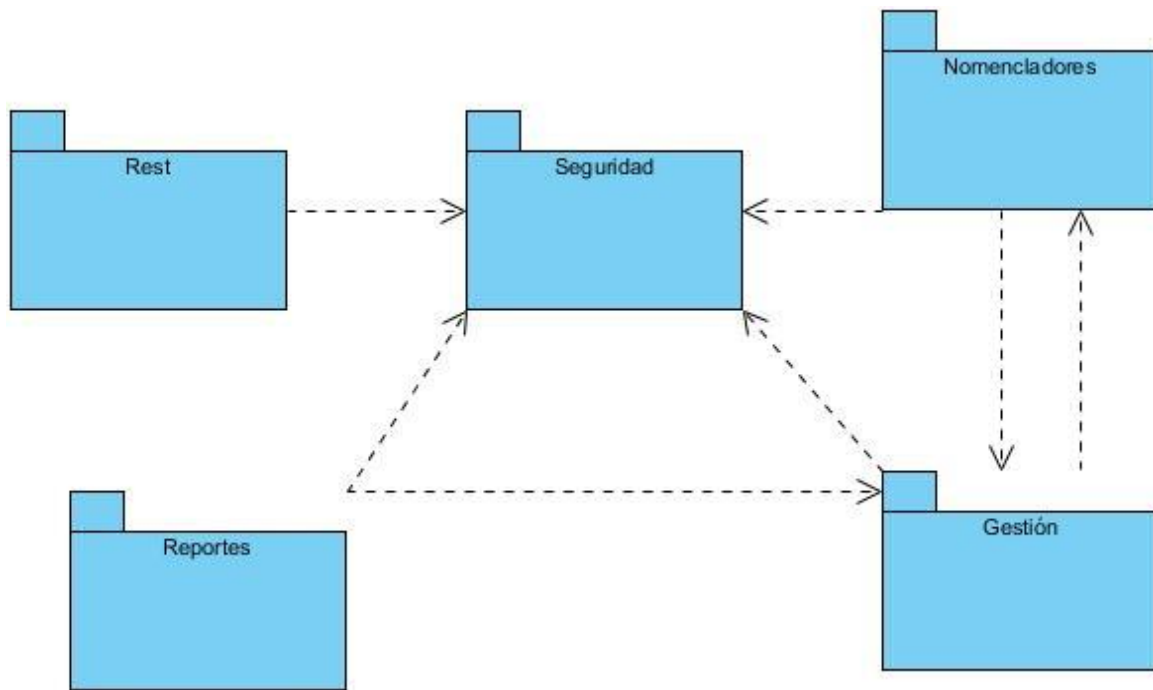


Ilustración 7 Paquetes y sus relaciones

Paquete Seguridad: Agrupa los casos de uso que intervienen en la seguridad de la aplicación y las funcionalidades necesarias para el acceso de usuarios al sistema, además de las funcionalidades de salva y restaura de la base de datos.

Paquete Nomencladores: Reúne los nomencladores a definir por el WebMaster.

Paquete Gestión: Agrupa todos los casos de uso relacionados con la gestión de los hospedajes, comentarios, rating, suscripción, etc.

Paquete Reportes: En este paquete se ejecutan las funcionalidades que permiten la obtención de reportes.

Paquete Rest: Agrupa la comunicación y traspaso de información entre la aplicación para teléfonos inteligentes con sistema operativo Android y el software.

3.6 Diagrama de casos de uso del sistema a automatizar.

Paquete Seguridad

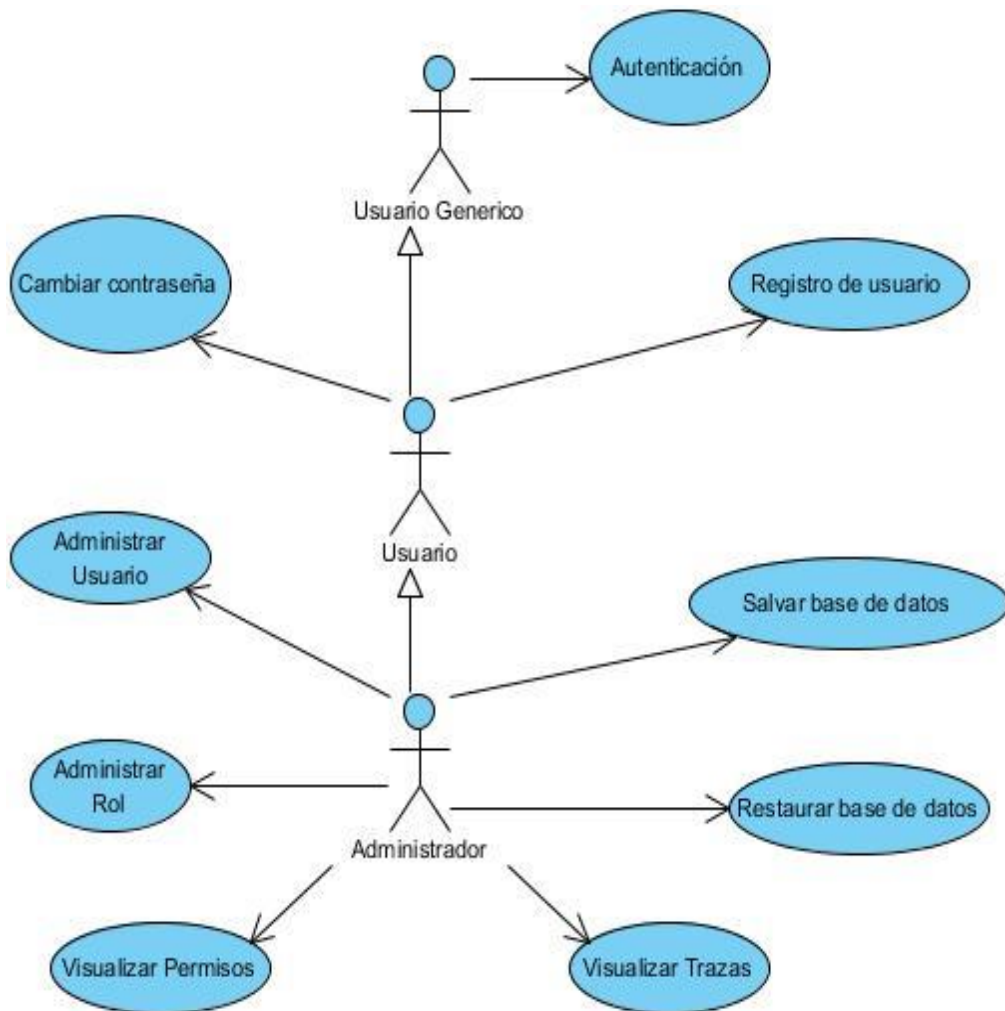


Ilustración 8 Diagrama de Caso de Uso del Paquete Configuración

Paquete de Nomencladores

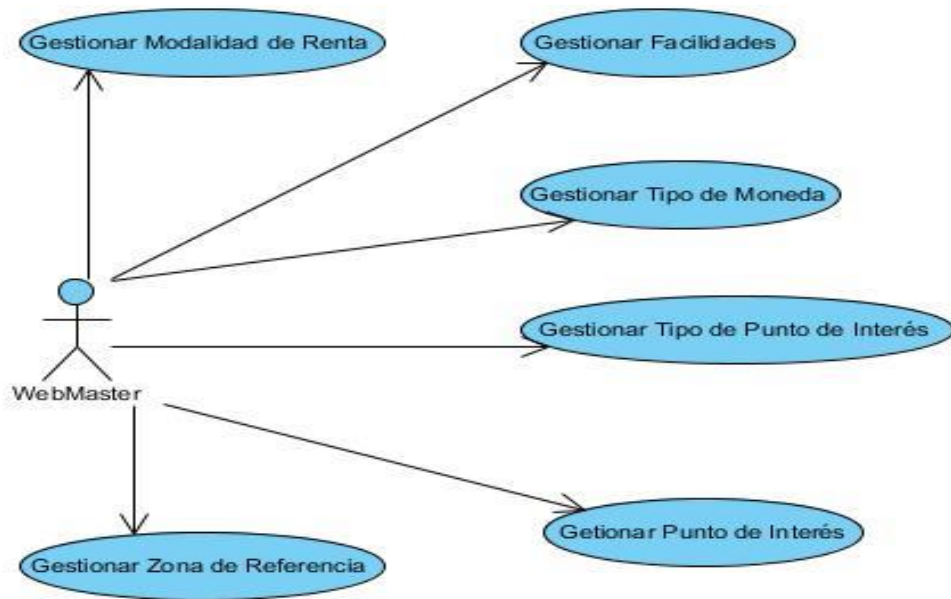


Ilustración 9 Diagrama de Casos de Uso Paquete Nomencladores

Paquete Gestión

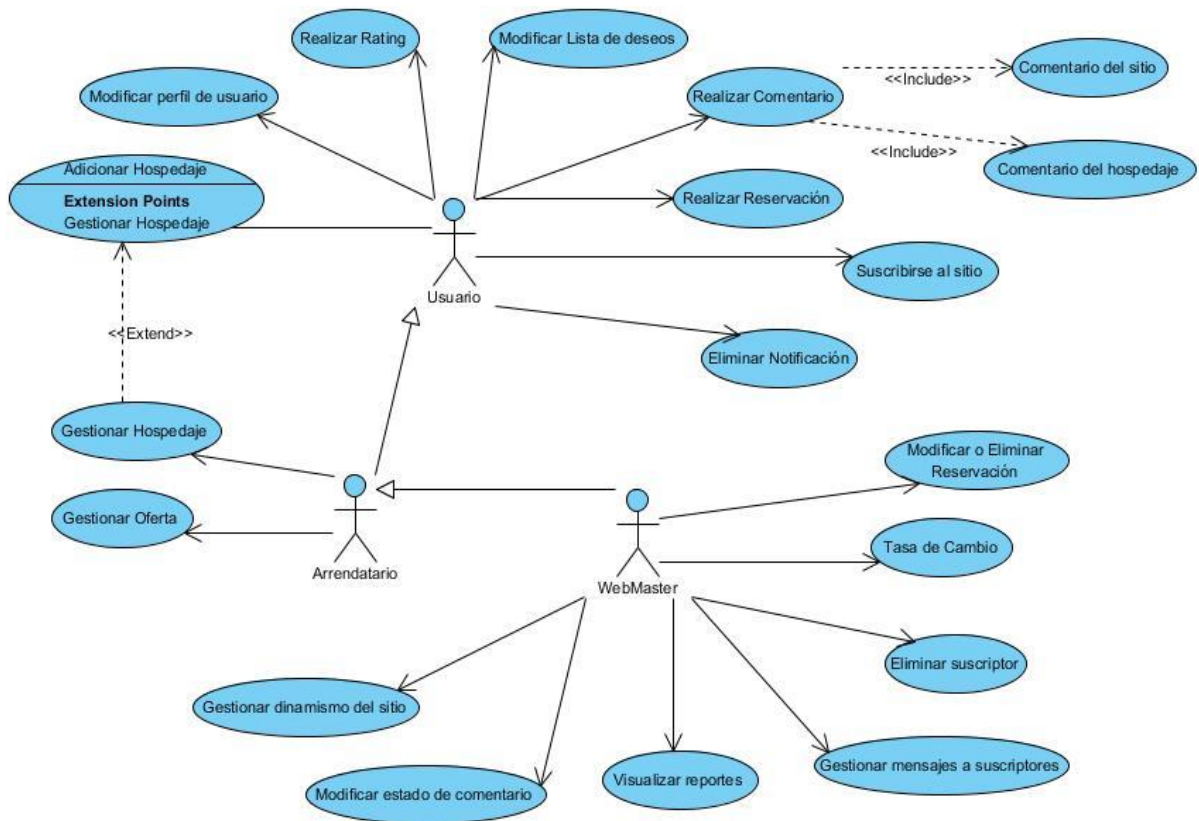


Ilustración 10 Diagrama de Casos de Uso Paquete Gestión

Paquete Reportes

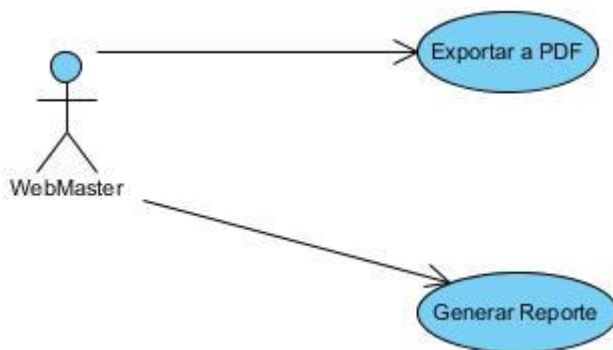


Ilustración 11 Diagrama de Casos de Uso del Paquete Reportes

Paquete Rest

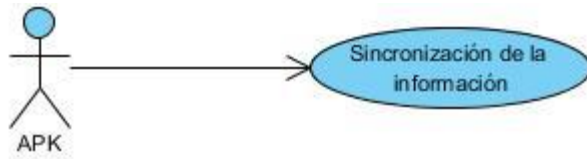


Ilustración 12 Diagrama de Casos de Uso del Paquete Rest

3.7 Descripción de los casos de uso.

3.7.1 Paquete Seguridad

Tabla 3 Descripción del caso de uso <Autenticación>

Nombre del caso de uso	Autenticación
Actores	Usuario
Resumen	Se inicia cuando el usuario quiere identificarse en el sistema. La aplicación verifica si los datos introducidos son correctos. En caso de que haya introducido algún dato incorrecto o de equivocarse 5 veces seguidas, se le muestra un mensaje de error al usuario, en el último caso, ya no podrá usar la autenticación. El caso de uso termina cuando el sistema le permite acceder a la página principal o cuando falla 5 veces.
Precondiciones	El usuario debe estar registrado en la base de datos.
Poscondiciones	Se muestra la información según el rol y los permisos que tenga.

Tabla 4 Descripción del caso de uso <Administrar Usuario>

Nombre del caso de uso	Administrar Usuario
Actores	Administrador
Resumen	El caso de uso se inicia cuando el Administrador desea insertar, modificar o desactivar un usuario. El sistema muestra un listado de los usuarios registrados, mostrando las opciones: insertar, modificar y desactivar. El caso de uso termina cuando se realiza una de las opciones.
Precondiciones	El administrador debe estar autenticado en el sistema.
Poscondiciones	Las acciones quedan guardadas en el historial del sistema.

Tabla 5 Descripción del caso de uso <Registro de usuario>

Nombre del caso de uso	Registro de Usuario
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario desea registrarse en el sistema. El sistema valida los datos requeridos y valida email. En caso de que los datos estén correctos, envía vía correo electrónico la <i>URL</i> de activación de usuario.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Poscondiciones	Las acciones quedan guardadas en el historial del sistema.

3.7.2 Paquete Nomencladores

Tabla 6 Descripción del caso de uso <Modalidad de Renta>

Nombre del caso de uso	Modalidad de Renta
Actores	WebMaster
Resumen	El actor puede agregar, modificar y eliminar una modalidad de renta. Al visualizar el listado de las modalidades las acciones de gestión están facilitadas en cada registro con botones representativos.
Precondiciones	El WebMaster debe estar autenticado.
Poscondiciones	Se realiza salva en el historial del sistema.

3.7.3 Paquete Gestión

Tabla 7 Descripción del caso de uso <Realizar Comentario>

Nombre del caso de uso	Realizar Comentario
Actores	Usuario
Resumen	Se inicia cuando el usuario se encuentra en la vista en detalles de un hospedaje o en la página de contactos del sitio. El usuario postea su comentario con una un estado de emoción.
Precondiciones	El usuario debe estar autenticado en el sistema.
Poscondiciones	Las acciones quedan guardadas en el historial del sistema. Se notifica a el-los WebMaster(s) que tiene un nuevo comentario que revisar para activar o dejar inactivo.

Tabla 8 Descripción del caso de uso <Realizar una Reservación>

Nombre del caso de uso	Realizar una Reservación
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario se encuentra ubicado en la página de detalles de un hospedaje y da un clic en reservación. Aparece una ventana para ingresar los datos requeridos o no. Se validan los campos y se envía la reservación vía correo electrónico al arrendatario.
Precondiciones	El usuario debe estar autenticado en el sistema.
Poscondiciones	Las acciones quedan guardadas en el historial del sistema. En caso de error en el envío del correo por problemas del servidor SMTP, el WebMaster puede ejecutar el reenvío posteriormente.

Tabla 9 Descripción del caso de uso < Gestionar mensajes a suscriptores>

Nombre del caso de uso	Gestionar mensajes a suscriptores
Actores	WebMaster
Resumen	El caso de uso se inicia cuando el actor se encuentra en la vista del listado de los mensajes a los suscriptores. El usuario puede agregar, modificar, eliminar cada mensaje. Una vez creado puede rectificar antes de enviar, cuando se envía cambia de estado a “enviado”.
Precondiciones	El WebMaster debe estar autenticado en el sistema. Deben existir al menos un suscriptor.
Poscondiciones	Las acciones quedan guardadas en el historial del sistema. En caso de error en el envío del correo por problemas del servidor SMTP, el WebMaster puede ejecutar el reenvío posteriormente.

3.7.4 Paquete Reportes

Tabla 10 Descripción del caso de uso < Exportar a PDF>

Nombre del caso de uso	Exportar a PDF
Actores	WebMaster
Resumen	Se inicia cuando el usuario se encuentra en la vista en detalles de uno de los cuatro reportes que ofrece el sistema. El actor presiona el botón “PDF” y acto seguido se descarga el documento.
Precondiciones	El WebMaster debe estar autenticado en el sistema
Poscondiciones	Para visualizar el reporte en formato pdf debe abrir el fichero descargado.

Tabla 11 Descripción del caso de uso < Generar Reporte>

Nombre del caso de uso	Generar Reporte
Actores	WebMaster
Resumen	El caso de uso se inicia cuando el usuario se encuentra en la vista administrativa del sistema. Dependiendo del reporte elegido, el sistema realiza la gestión lógica.
Precondiciones	El WebMaster debe estar autenticado en el sistema. La base de datos debe tener registros guardados para realizar el análisis.
Poscondiciones	Muestra uno o varios gráficos más un listado con el resultado.

3.7.5 Paquete Rest

Tabla 12 Descripción del caso de uso < Sincronización de la información>

Nombre del caso de uso	Sincronización de la información
Actores	APK
Resumen	Conjunto de actividades que ejecutan la sincronización de la base de datos de la aplicación para teléfonos inteligentes.
Precondiciones	La aplicación debe tener acceso a internet.
Poscondiciones	Se actualizan los registros de la base de datos del servidor y de cada base de datos instalada en la APK.
Requisitos especiales	Se debe tener una velocidad de conexión a internet por encima de 20 Kbit/s.

3.8 Definición de los requisitos no funcionales.

Apariencia o interfaz externa.

La aplicación debe tener una interfaz clara, sencilla y que brinde solo la información oportuna, el sistema mostrará el nombre de la empresa en toda página y documento emitido; además se siguieron las pautas de diseño para la interfaz gráfica de las aplicaciones informáticas de la entidad.

Usabilidad.

El sistema a está dirigido a usuarios con una experiencia básica en el trabajo con las Tics por lo que deberá ser fácil de usar.

Rendimiento.

La aplicación debe garantizar la disponibilidad de la información de manera ágil realizando consultas a la base de datos, mejorando el tiempo de búsqueda.

Portabilidad.

La aplicación será desarrollada en el lenguaje **Python**, siendo este un lenguaje multiplataforma, lo que permitirá a la aplicación ejecutarse en estaciones de trabajo o servidores con sistemas operativos Microsoft Windows o Unix/Linux sin grandes cambios en el código fuente.

Seguridad

La administración de la aplicación será accedida por usuarios con permisos otorgados por el administrador del sistema, por lo que se garantizará un nivel de seguridad y protección a la información adecuado, quedando solo disponible según los roles de cada usuario. Todas las acciones de los usuarios serán registradas en la base de datos permitiendo identificar un error en el tratamiento de la información. El sistema permitirá realizar salvadas y restauraciones de la base de datos para garantizar una rápida recuperación en caso de hechos extraordinarios.

Político-culturales.

El sistema estará disponible en el idioma español.

Legales.

El sistema debe contar con una política de privacidad, pues maneja datos personales del usuario.

Ayuda y documentación en línea.

El sistema debe contar con una ayuda para aquellos que quieran viajar a Cuba y hospedarse en una casa, así como también instrucciones para todo aquel que quiera inscribirse en la ONAT como arrendatario.

Software.PC-Usuario:

- Sistema Operativo: Windows o Linux
- Navegador web

Servidor web:

- Sistema Operativo: Windows o Linux
- Servidor web Apache.
- Python 3.7 o superior.
- Dependencias de Python

Servidor de base de datos:

- Sistema Operativo: Windows o Linux
- PostgreSQL

Hardware.

Las Estaciones de trabajo deberán tener al menos 512 MB de RAM, Micro Procesador 2.0 GHz y 5 GB de HDD. El servidor de base de datos y el servidor web deberán tener 4 GB de RAM o superior, Micro Procesador 2.5 GHz– y 10GB de HDD.

3.9 Conclusiones.

- a) Se definieron los requisitos funcionales y no funcionales de la aplicación, según procesos a automatizar definidos con el cliente.

- b) Se describieron los Casos de Uso del Sistema según requisitos funcionales, agrupándolos para un mejor diseño y desarrollo en paquetes.
- c) Se identificaron los paquetes y sus relaciones, identificando la relación del paquete de seguridad, con el resto de los paquetes definidos.

Capítulo 4 Descripción de la solución propuesta

4.1 Introducción.

El diseño es una parte importante dentro del proceso de desarrollo de software. Durante el mismo se debe velar por el cumplimiento de los requerimientos que se definen en un sistema. Por ello, en este capítulo se utilizarán algunos artefactos de UML como son: el diagrama de clases del diseño, el modelo lógico y el modelo físico de datos, que son la base para construir la base de datos que soportará el sistema. Además, se elaborará el diagrama de despliegue y se enunciarán los principios fundamentales de diseño que se seguirán en el desarrollo de la solución propuesta.

4.2 Diagrama de clases del diseño.

4.2.1 Paquete Seguridad

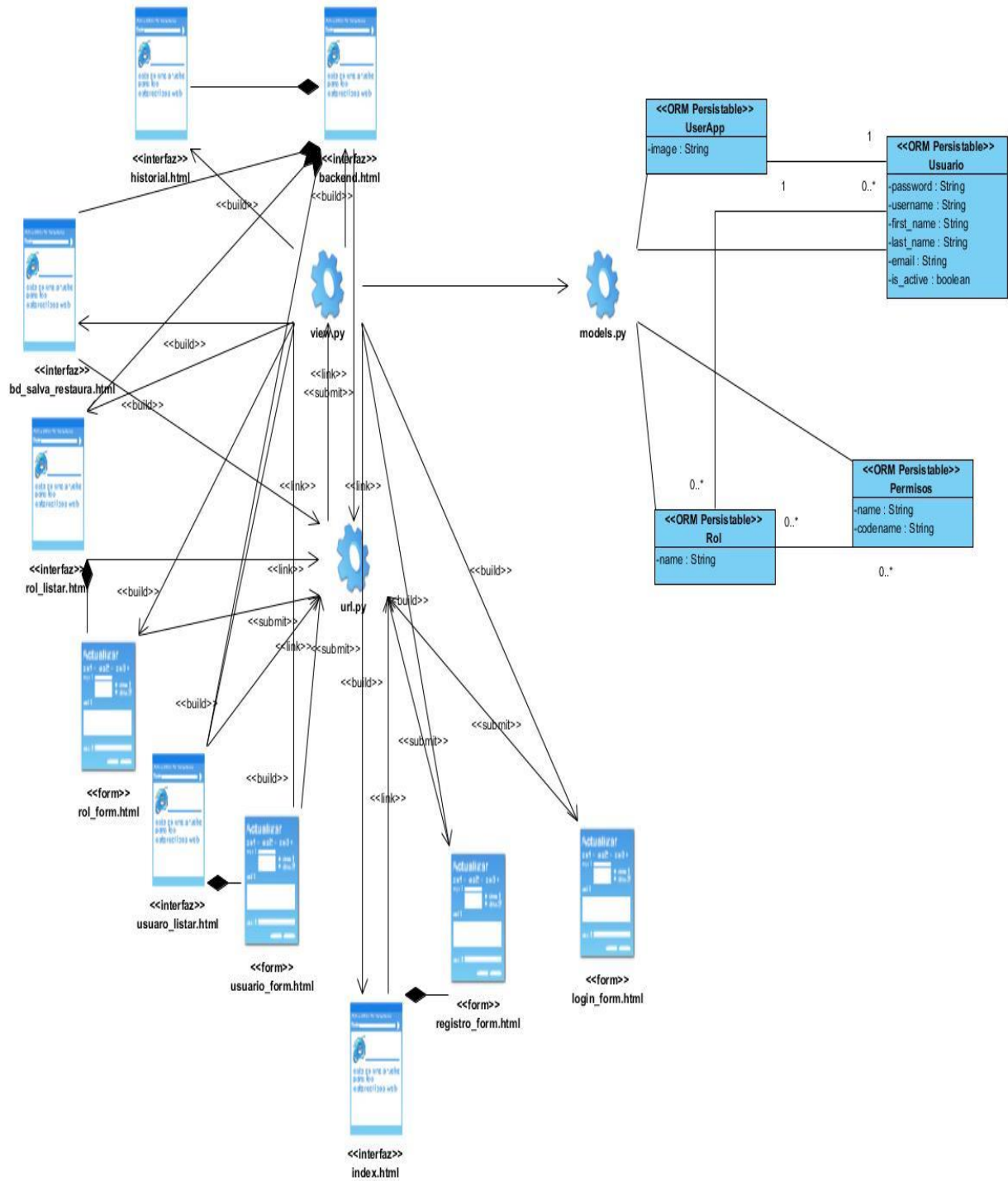


Ilustración 13 Diagrama de clases del paquete Seguridad

4.2.2

Paquete Nomencladores

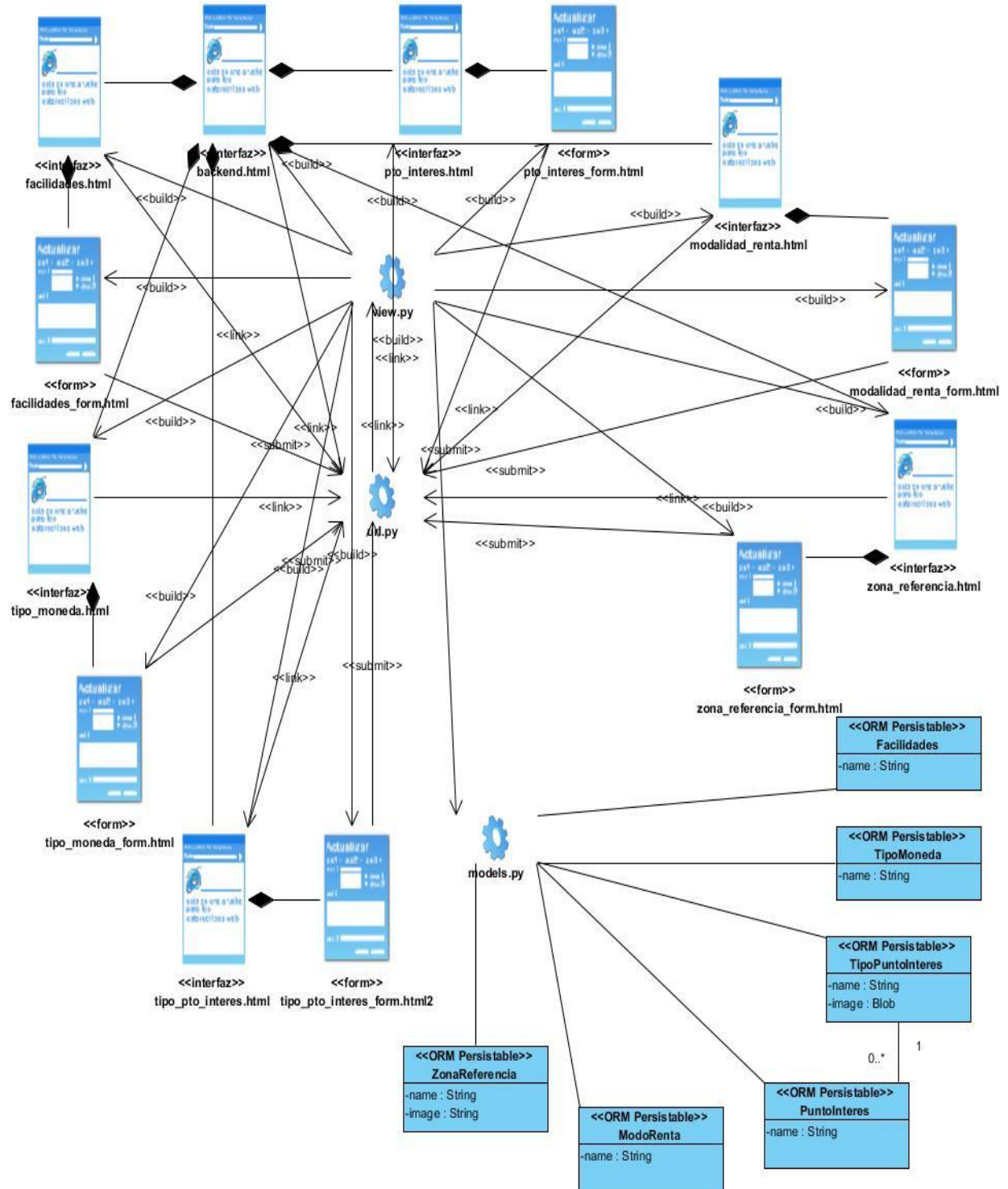


Ilustración 14 Diagrama de clases del paquete Nomencladores

4.2.3 Paquete Reportes

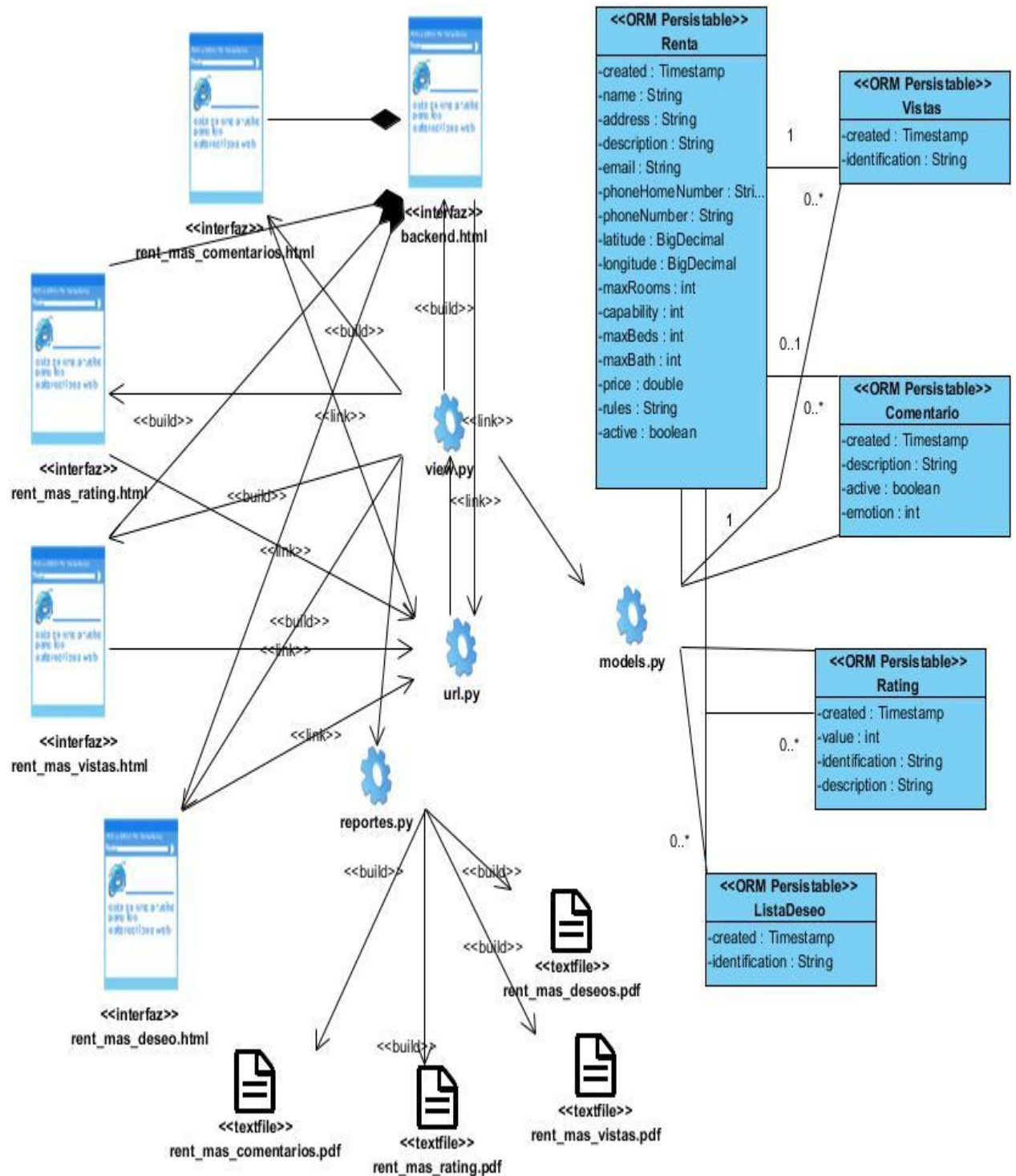


Ilustración 15 Diagrama de clases del paquete Reportes

4.2.4 Paquete Gestión

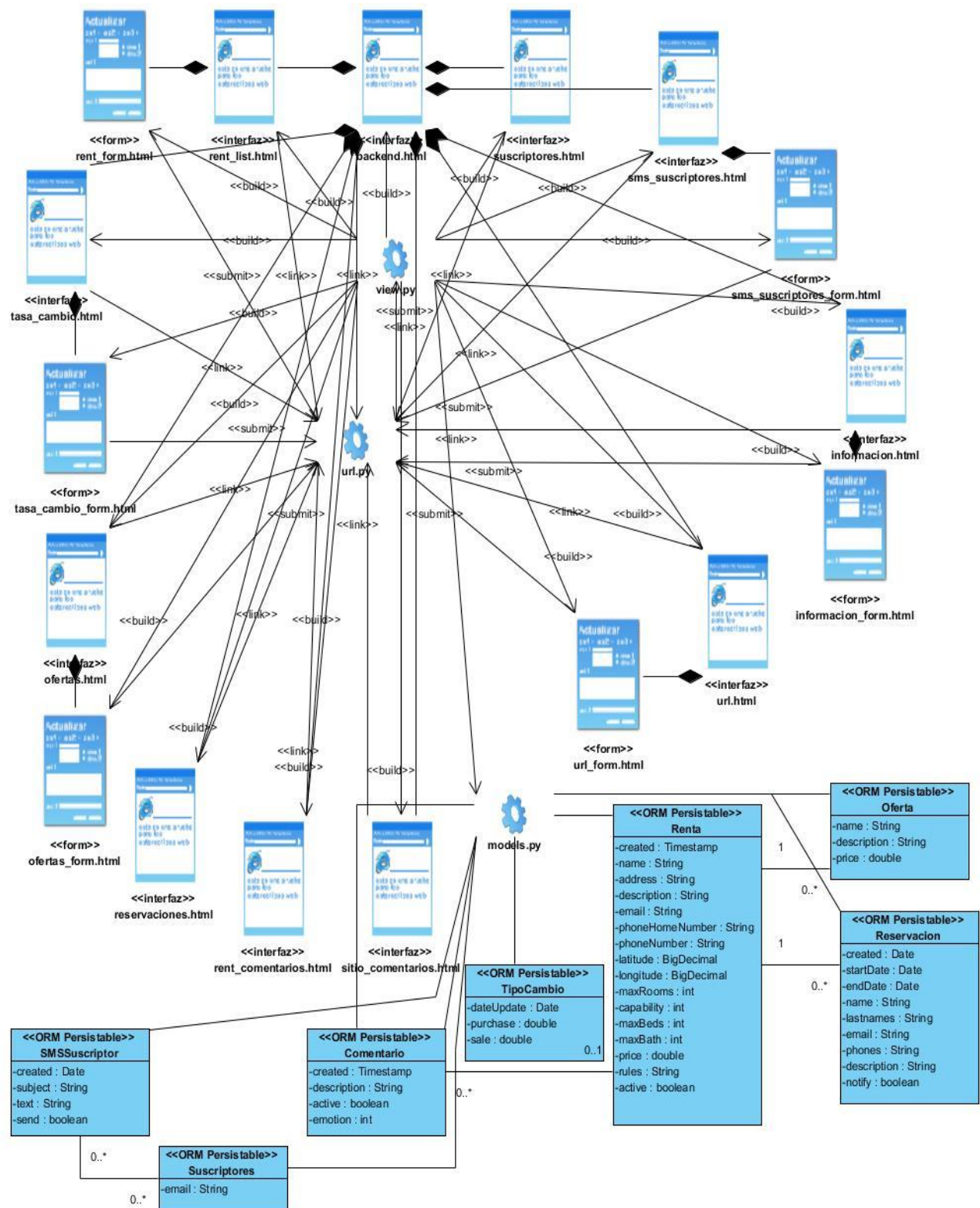


Ilustración 16 Diagrama de clases del paquete Gestión (primera parte)

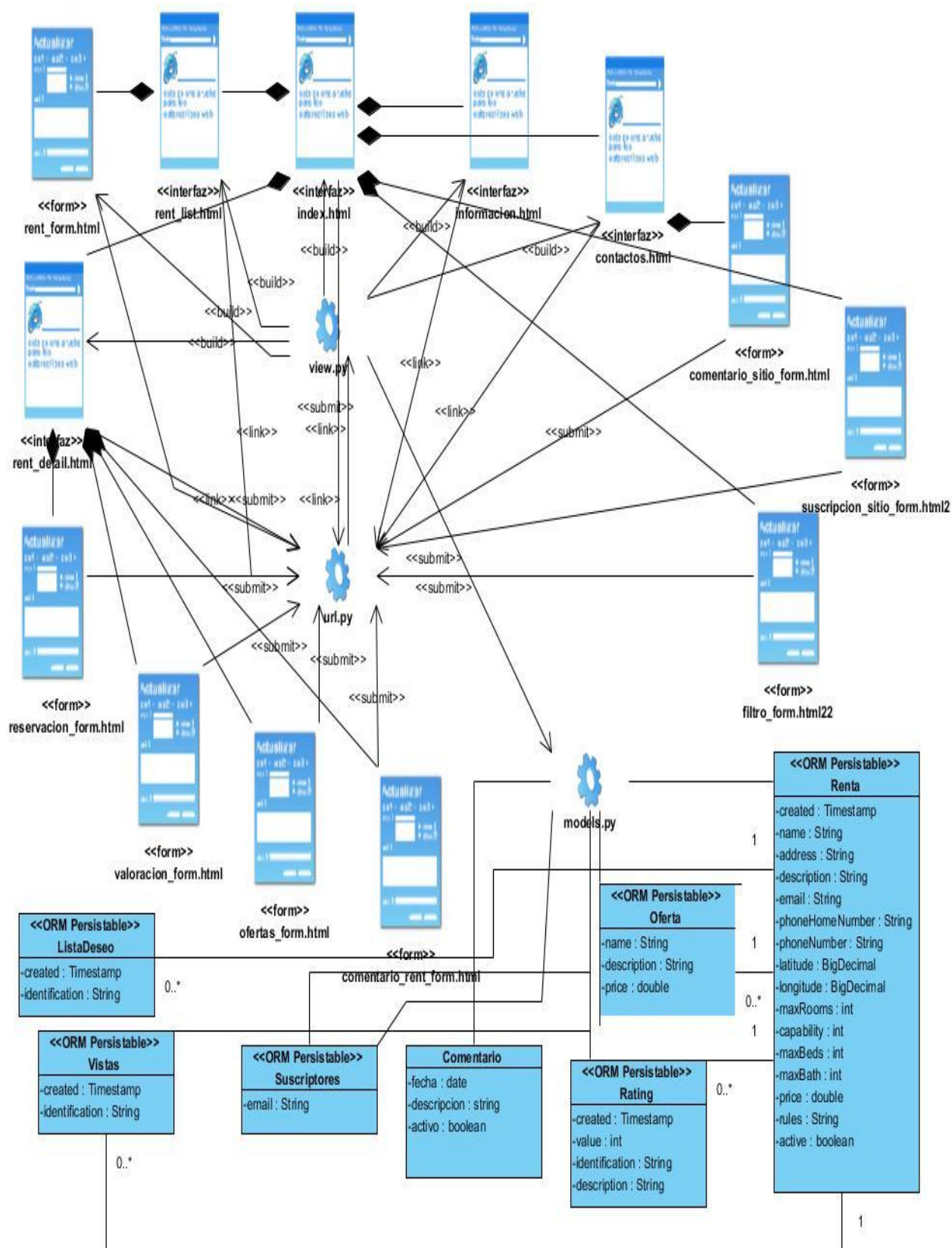


Ilustración 17 Diagrama de clases del paquete Gestión (segunda parte)

4.2.5 Pacote *Rest*

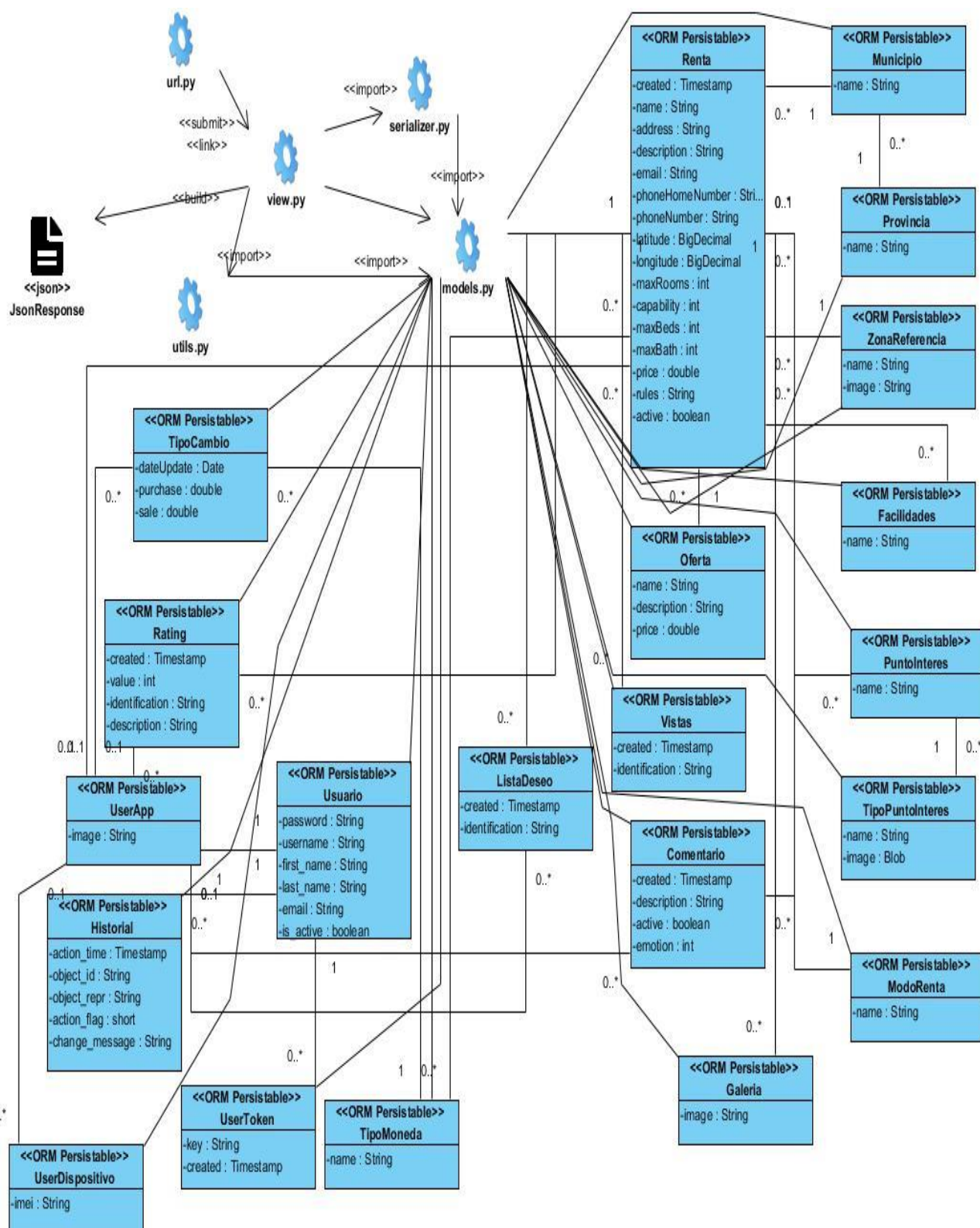


Ilustración 18 Diagrama de clases del paquete *Rest*

4.3 Diseño de la base de datos.

4.3.1 Modelo lógico de datos.

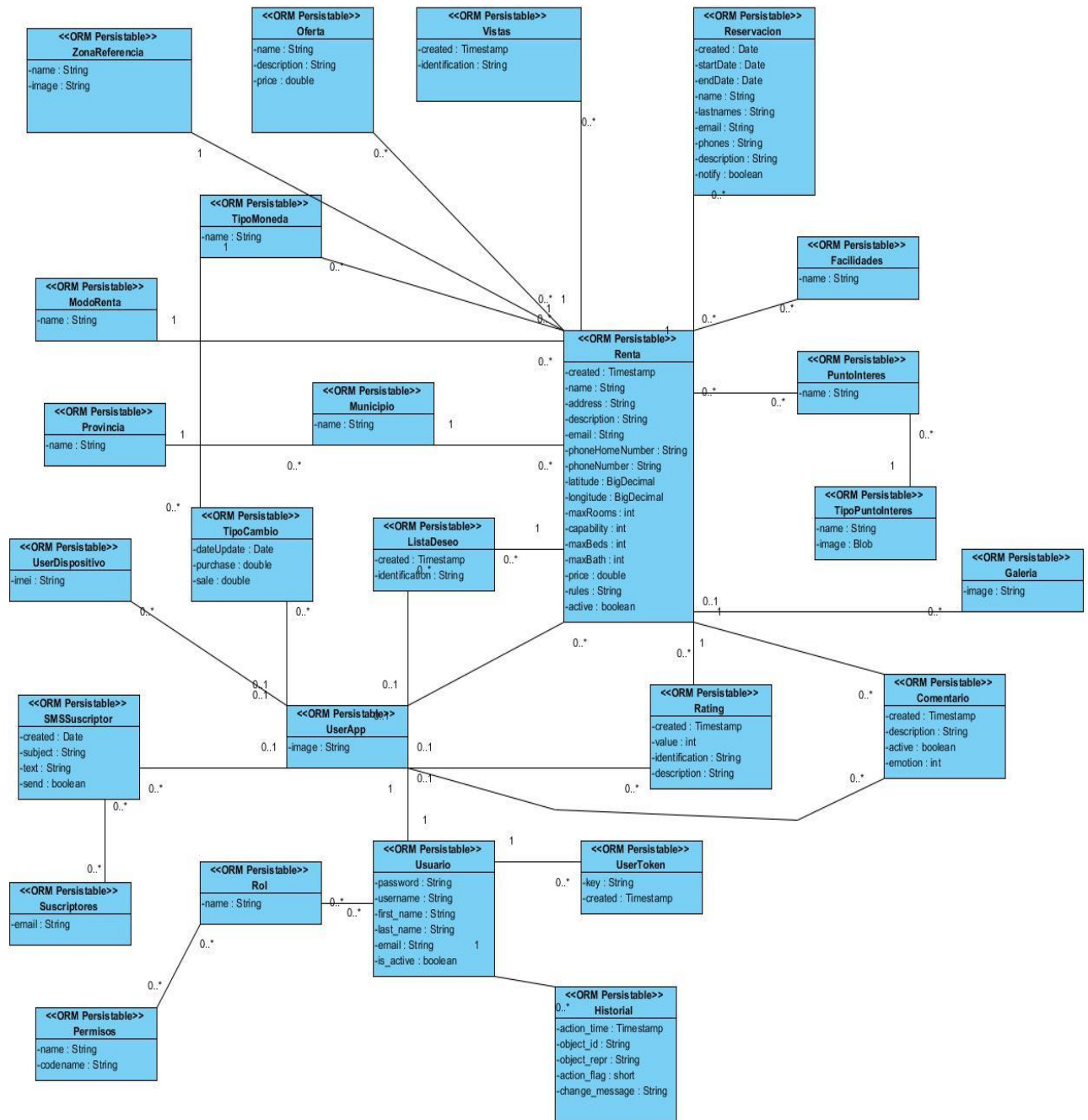


Ilustración 19 Modelo lógico de datos

4.3.2 Modelo físico de datos.

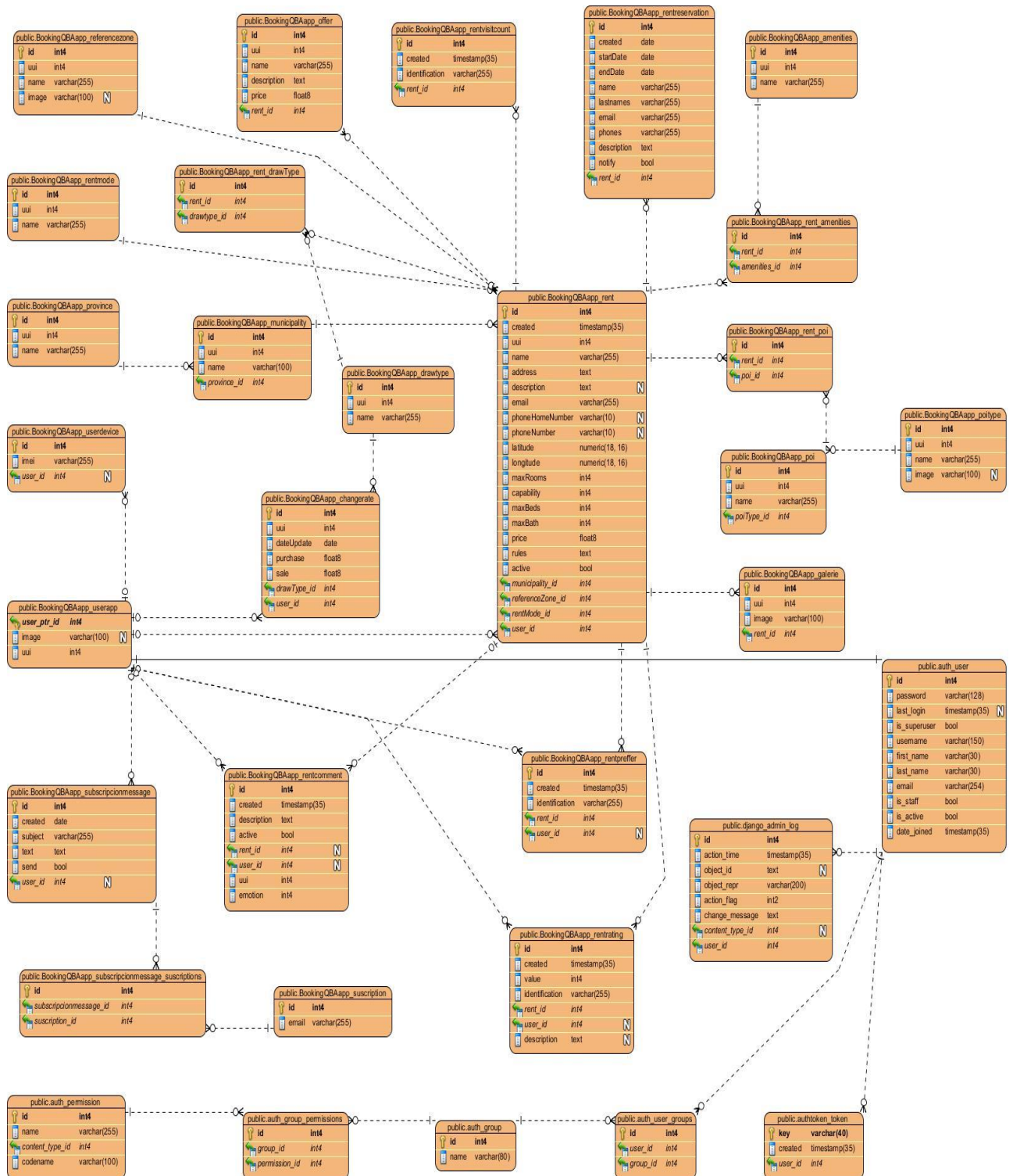


Ilustración 20 Modelo físico de datos

4.4 Principios de diseño.

4.4.1 Interfaz de usuario.

Se tuvo en cuenta en el desarrollo de la interfaz del sistema propuesto, la estabilidad y uniformidad del diseño. Se emplean fundamentalmente los cuatro colores representativos definidos para la aplicación y para APK. Se consideró utilizar una misma tipografía, forma y estilo en todas las páginas informativas como también con el sitio de administración. El sistema muestra un ambiente serio y profesional.

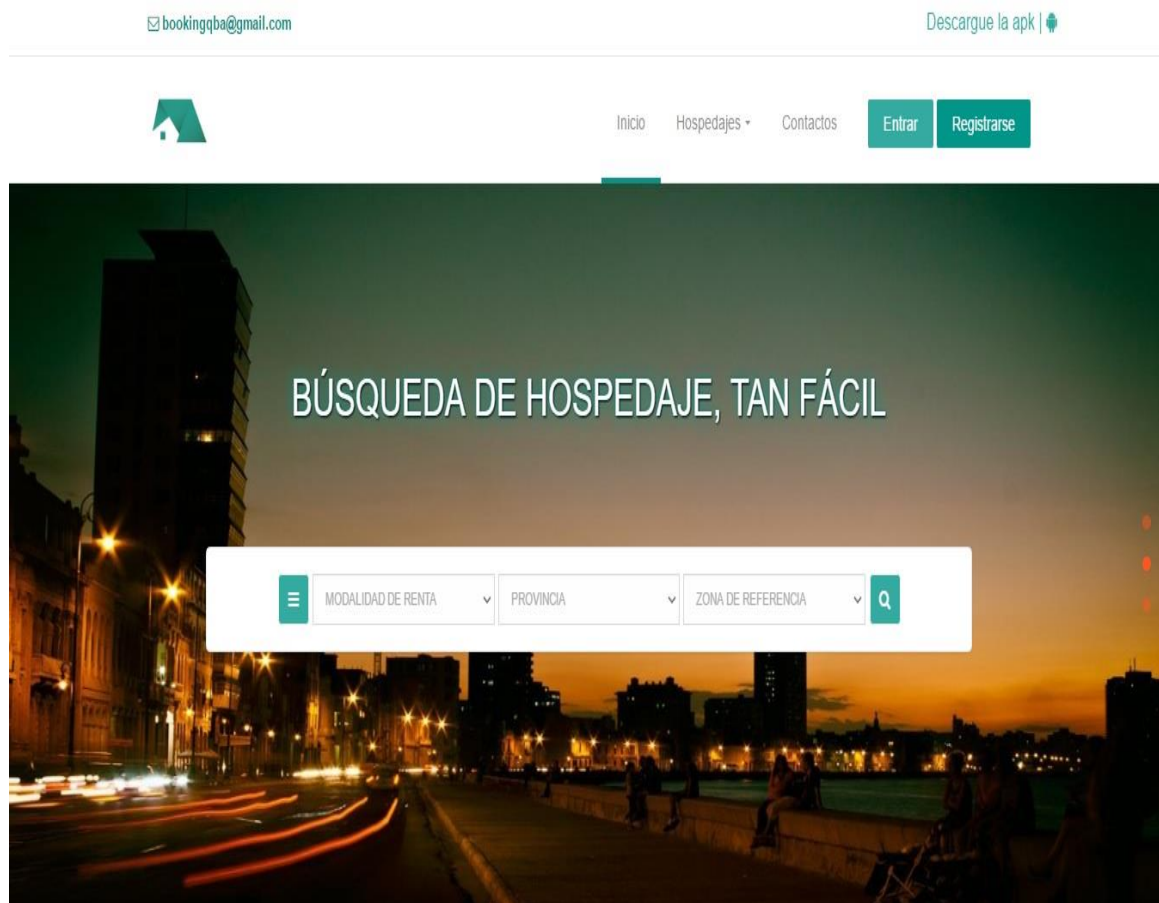


Ilustración 21 Interfaz de usuario (front)

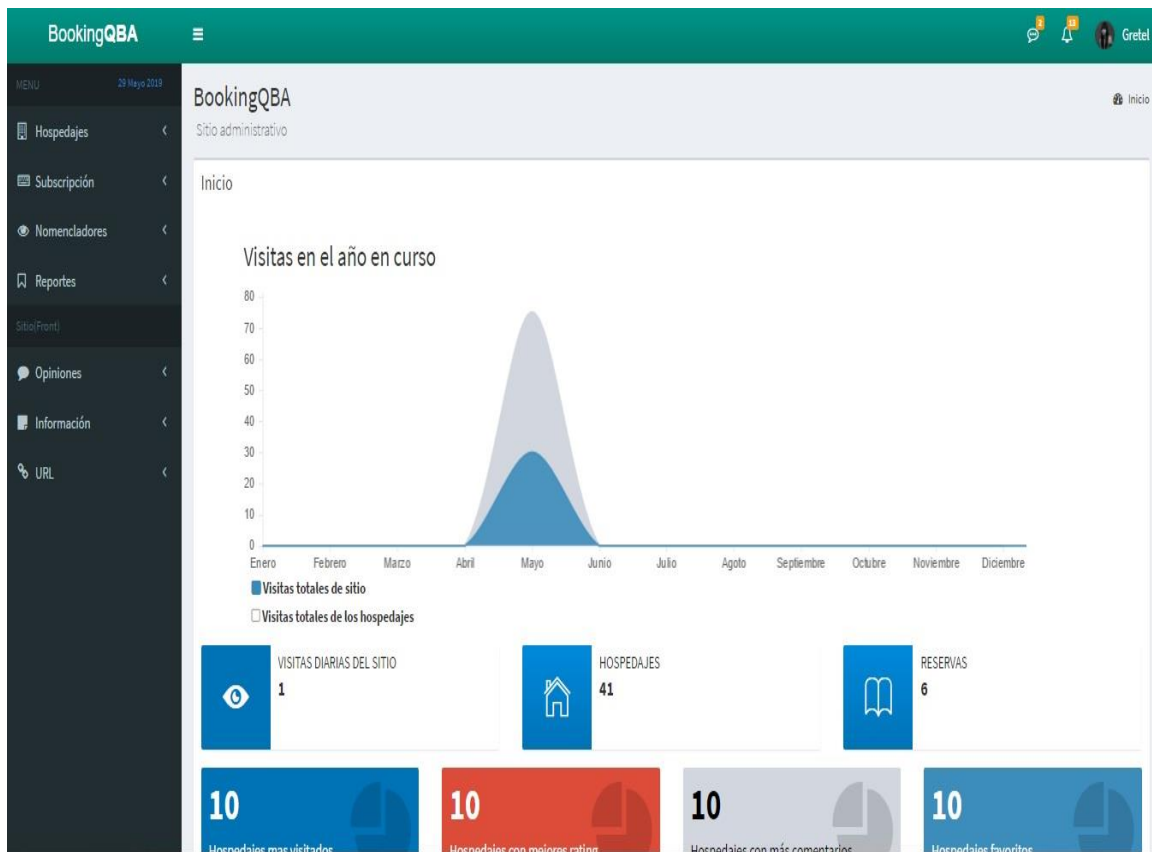


Ilustración 22 Interfaz de usuario (administración)

4.4.2 Formato de salida de los reportes.

Teniendo presente que la solución propuesta es Web, las informaciones a mostrar se conciben sobre ventanas de la misma, con la utilización de un formato de letra claro y legible. Los reportes de tipo informe presentan la opción exportar en formato PDF, siguiendo una estructura estándar, por ejemplo:



Hospedajes con más visitas

NO.	HOSPEDAJE	VISITAS
1	Casa El Encanto	79
2	Casa Eduardo y Zoe	25
3	Casa Bernardo Ortiz	14
4	Palacio del jardín in downtown	12
5	Cojimar See	12
6	Casa Margarita	8
7	Apartamento Ana	7
8	Casa Doris	7
9	Hostal Punta La Piedra	6
10	Pegasus	5
11	Casa Reyes	3
12	Agroturismo "La Campiña" Cabaña	2
13	Private entrance great location	2
14	Casa Alta	2
15	Mi casa verde	1
16	Villa Vigia	1
17	Apartamento céntrico en Holguín	1
18	CASA BELLA VISTA	1
19	Hostal Villa Manuela	1
20	Villa Amelia	1
21	Casa Pepe Irrragor	1

Generado a las: 03-06-2019 a las 11:11:41AM

Ilustración 23 Reporte en formato PDF

4.4.3 Ayuda.

La aplicación ofrece información general a modo de ayuda para los usuarios arrendatarios y para los usuarios comunes que visitan el sitio. Esta contiene:

- ONAT Información
- Arrendatarios
- Tasa de Cambio

- Manual de viaje
- Ayuda

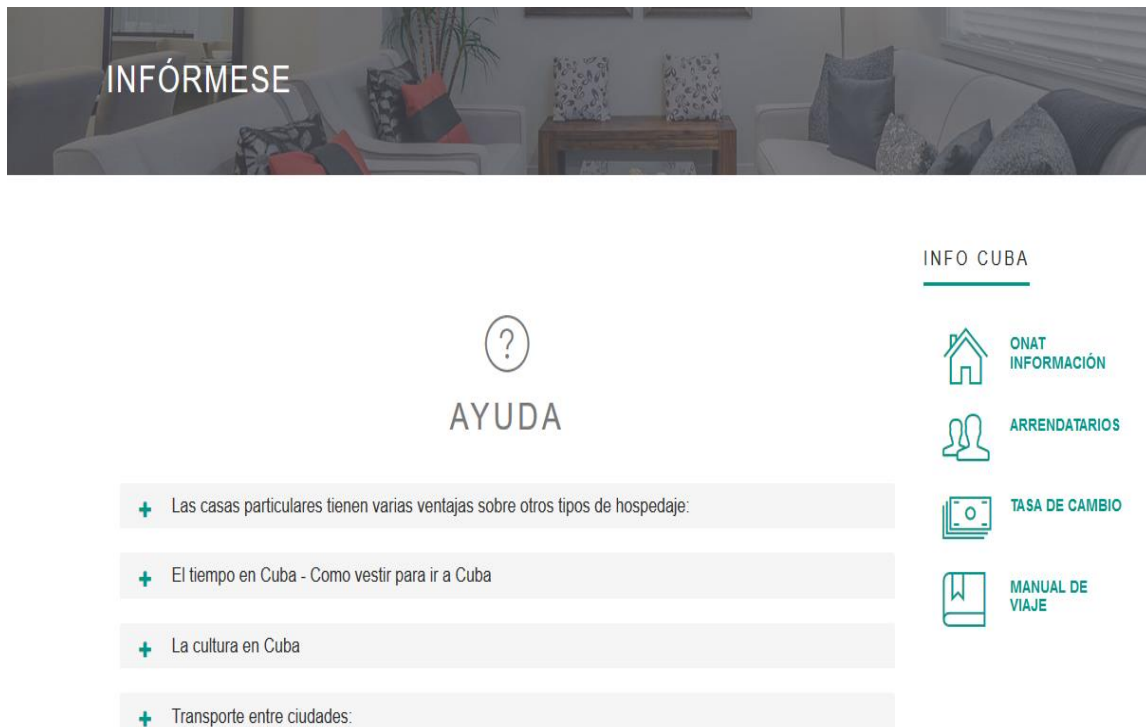


Ilustración 24 Ayuda del sistema (front)

4.5 Patrones de diseño

En el desarrollo de una aplicación informática, la solución de problemas de características similares, en distintos módulos, secciones, etc., es una realidad a la que se enfrentan los programadores comúnmente. En aras de mejorar la capacidad de darle solución a estos, muchas veces pequeños, pero muy recurrentes problemas, surgen los patrones de diseño a partir de soluciones ya probadas. De esta forma cada patrón tiene una descripción de la solución, según el problema a tratar, permitiendo emplear la misma solución sin implementarla nuevamente. A continuación, se exponen los principales patrones presentes en la solución implementada.

Patrones GRASP (General Responsibility Assignment Software Patterns, por sus siglas en inglés: Asignación general de responsabilidad, patrones de software):

Describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Su utilidad viene a la hora de caracterizar objetos y clases. Saber dónde ubicar una responsabilidad (un servicio), quién es responsable de crear objetos, manejar eventos[28].

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Los formularios de la aplicación heredan de la clase *ModelForm*, la que se encarga de crear el objeto; esta clase está implementada en el *framework*. Un ejemplo a continuación.

```
def referencezone_create(request):
    if request.POST:
        form = forms.ReferenceZoneForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            id_reference = ReferenceZone.objects.last()
            register_logs(request, ReferenceZone, id_reference.uui, id_reference.__str__(), 1)
            sweetify.success(request, "Éxito creando la zona de referencia")
            return HttpResponseRedirect('/administration/nomenclator/referencezone/list')
        else:
            sweetify.error(request, "Error en el formulario")
    else:
        form = forms.ReferenceZoneForm()
    args = {}
    args['form'] = form
    return render(request, 'bookingbaapp/referencezone_form.html', args)
```

Ilustración 25 Función que crea el objeto

```
class ReferenceZoneForm(forms.ModelForm):
    class Meta:
        model = models.ReferenceZone
        fields = "__all__"
        widgets = {
            "name": widgets.TextInput(attrs={'class': 'form-control'}),
            "image": widgets.ClearableFileInput(attrs={'class': 'form-control'}),
        }
```

Ilustración 26 Clase que hereda de ModelForm

Experto

El patrón Experto en Información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Se evidencia en la clase *ReferenceZone*, la misma contiene toda la información que comprende a una zona de referencia.

```
class ReferenceZone(models.Model):
    uui = models.UUIDField(default=uuid.uuid4, editable=False)
    name = models.CharField(max_length=255, verbose_name="Nombre", unique=True)
    image = models.ImageField(upload_to='static/upload/galeria', verbose_name="Imagen", null=True)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name_plural = "Zona Referencial"
```

Ilustración 27 Patrón Experto reflejado en la clase *ReferenceZone*

Gang-of-Four (GoF, en español Banda de los Cuatro):

Describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. Se clasifican según su propósito en patrones de creación, estructurales y de comportamiento; y según su ámbito: en clase y objeto[29].

Decorador

Añade nuevas responsabilidades a un objeto, proporciona una alternativa flexible a la herencia para extender funcionalidad. Python maneja este patrón agregando funcionalidades al inicio o al final del método decorado. Un ejemplo se evidencia en el siguiente código, donde se requieren permisos específicos para ejecutar el mismo.

```
@permission_required('BookingOBApp.add_referencezone')
def referencezone_list(request):
    referencezones = ReferenceZone.objects.all()
    return render(request, 'Admin/Nomenclator/reference_zone.html', {'referencezones': referencezones})
```

Ilustración 28 Patrón Decorator reflejado en método *referencezone_list*

4.6 Tratamiento de errores.

Los tratamientos de errores se realizan con el propósito de lograr un producto con calidad, que se caracterice por brindar un ambiente de trabajo estable y seguro al usuario. El diseño de los formularios de entrada de datos presentes en la aplicación se elaboró de una manera clara y legible, utilizando siempre que fuera posible, componentes de selección y marcado, para disminuir el margen de error. Se valida que los datos sean consistentes y que no falte ninguna información imprescindible para la ejecución de la acción, en caso de que el usuario intente visitar una URL que no tiene permiso automáticamente irá al error 404. Seguidamente se muestran ejemplos de validación con sus respectivos mensajes.

Empecemos con la información básica

Foto Portada

CLICK AQUI

(requerido)

Nombre: (requerido)

Campo requerido.

Dirección: (requerido)

Campo requerido.

Precio: (requerido)

Campo requerido.

Ilustración 29 Error en los campos requeridos



Ilustración 30 Error de URL

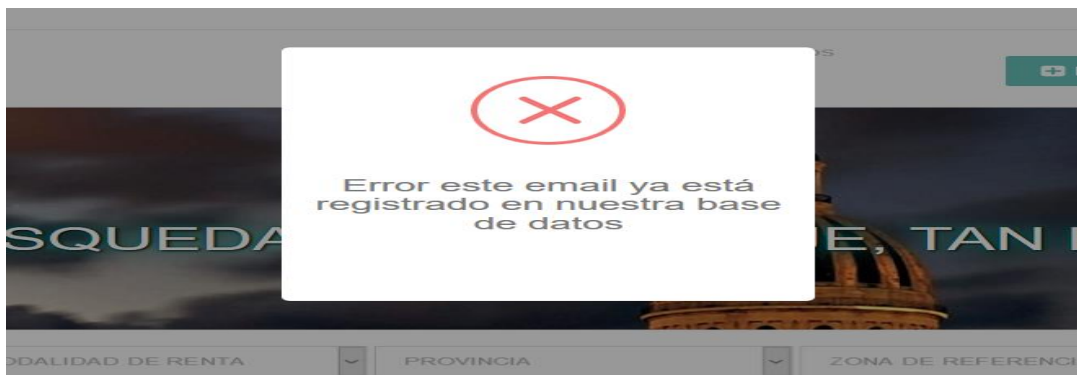


Ilustración 31 Error validación de datos desde el servidor

4.7 Diagrama de despliegue.

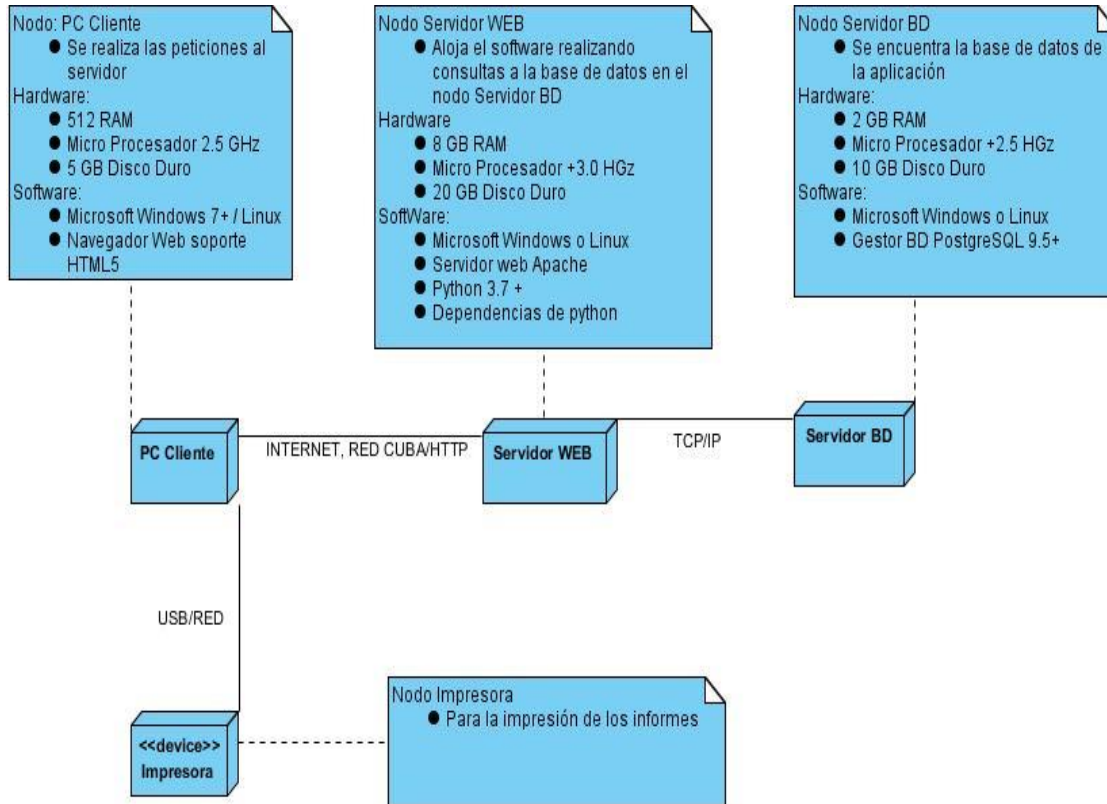


Ilustración 32 Diagrama de despliegue

4.8 Estructuración en capas.

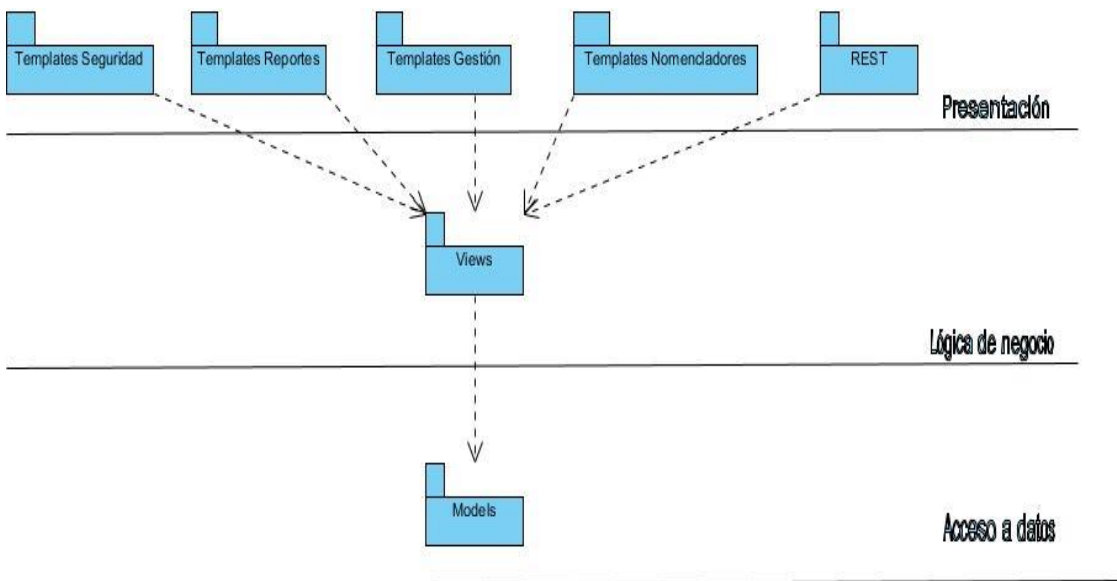


Ilustración 33 Estructuración por capas

4.9 Conclusiones

- a) Se definieron los principios de diseño que garantizan el cumplimiento de los requisitos no funcionales.
- b) Se aplicaron los patrones de diseño GRASP y GOF para describir los principios fundamentales de diseño de objetos para la asignación de responsabilidades y soluciones a los problemas definidos en el proyecto.
- c) Se delinearon los diagramas de clases de diseño para el desarrollo del sistema favorable a la vista del programador.
- d) Se diseñó la estructura lógica y física de la base de datos, lo que permite el almacenamiento de forma persistente y por tipo de datos según necesidades propias del sistema.
- e) Se esbozó el diagrama de despliegue de la solución desarrollada, cumpliendo las características definidas según requisitos no funcionales.

Capítulo 5 Validación y factibilidad de la solución propuesta.

5.1 Introducción al capítulo.

En este Capítulo se abordará el proceso de validación realizado sobre la aplicación, definiendo las pruebas efectuadas y el porqué de dichas pruebas, mostrando además los resultados obtenidos. También se estudiará la factibilidad del proyecto evaluando los costos de desarrollo de la aplicación. Se realizará una estimación del esfuerzo, el tiempo de desarrollo y el costo total de la realización de la aplicación.

5.2 Pruebas automatizadas al sistema.

Las pruebas de software son una de las partes más importantes de cualquier proyecto de software, ya que aporta calidad y seguridad al código[30].

Un conjunto de actividades de pruebas suele orientarse a comprobar determinados aspectos de un sistema software (o de una parte del mismo).[31]

Las **pruebas automatizadas** son una herramienta extremadamente útil para eliminar errores en el desarrollo de sitios web modernos. Se puede utilizar una colección de pruebas para resolver o evitar una serie de problemas:

- Cuando está escribiendo un nuevo código, se pueden usar pruebas para validar que el código funciona como se esperaba.
- Cuando se está modificando el código anterior, se pueden usar pruebas para asegurarse que los cambios no hayan afectado el comportamiento de la aplicación de forma inesperada.

Como ejemplo en el proyecto se realizan dos tipos de pruebas automatizadas, la creación de un hospedaje donde intencionalmente aplicamos un error, así como cíclicamente se registran 100 hospedajes al instante. A continuación se muestra un ejemplo del código y el resultado.

```

from django.test import TestCase
from model_mommy import mommy
from BookingQBAapp import models

class RentCreateTestModel(TestCase):

    def setUp(self):
        municipality = models.Municipality.objects.all().first()
        rentMode = models.RentMode.objects.all().first()
        user = models.UserApp.objects.all().first()
        lon = -82.36856
        lat = 23.09117
        self.rent = models.Rent.objects.create(name="Hospedaje-fixture-test1-", address="Esquina tunico numero-test",
                                                slogan="esta es la mejor casa",
                                                description="lorem insu lorem insu lorem sino nidnd ndnsiidd esta esenrn idd",
                                                email="correo-test@gmail.com", price=15, rentMode=rentMode, municipality=municipality,
                                                user=user, active=True, rules='', longitude=_lon, latitude=_lat,
                                                capability=5, maxBeds=2, maxBath=2, maxRooms=3)

    def test_rent_creation_100(self):
        self.assertTrue(isinstance(self.rent), models.Rent)
        print(self.rent)

class Rent100CreateTestModel(TestCase):

    def setUp(self):
        self.rent = []
        for i in range(100):
            rent = mommy.make(models.Rent)
            self.rent.append(rent)

    def test_rent_creation_100(self):
        self.assertTrue(100, len(self.rent))

```

Ilustración 34 Pruebas automatizadas, código en el archivo test.py

```

System check identified no issues (0 silenced).
test_rent_creation_100 (BookingQBAapp.tests.Rent100CreateTestModel) ... ok
test_rent_creation_100 (BookingQBAapp.tests.RentCreateTestModel) ... ERROR

=====
ERROR: test_rent_creation_100 (BookingQBAapp.tests.RentCreateTestModel)
-----
Traceback (most recent call last):
  File "E:\escuela\Tesis\BookingQBA\BookingQBAapp\tests.py", line 18, in setUp
    capability=5, maxBeds=2, maxBath=2, maxRooms=3)
  File "C:\Program Files\Python37\lib\site-packages\django-1.11.20-py3.7.egg\django\db\models\manager.py", line 85, in manager_method
    return getattr(self, get_queryset(), name)(*args, **kwargs)
  File "C:\Program Files\Python37\lib\site-packages\django-1.11.20-py3.7.egg\django\db\models\query.py", line 392, in create
    obj = self.model(**kwargs)
  File "C:\Program Files\Python37\lib\site-packages\django-1.11.20-py3.7.egg\django\db\models\base.py", line 573, in __init__
    raise TypeError("%s' is an invalid keyword argument for this function" % list(kwargs)[0])
TypeError: 'slogan' is an invalid keyword argument for this function

-----
Ran 2 tests in 1.264s

FAILED (errors=1)
Destroying test database for alias 'default' ('test_bookingqbadb')...

```

Ilustración 35 Resultado de las pruebas automatizadas

5.3 Análisis estático del código

El análisis estático del código se refiere al proceso de evaluación del código fuente sin ejecutarlo, es en base a este análisis que se obtendrá información que nos permita mejorar la línea base de nuestro proyecto, sin alterar la semántica original de la aplicación[32].

Pylint es una herramienta que todo programador en Python debe considerar en su proceso de integración continua. Básicamente su misión es analizar código en Python en busca de errores o síntomas de mala calidad en el código fuente. Emite una valoración en base a 10 puntos al final de su análisis, la misma puede tener un valor negativo.

A continuación se muestra el resultado del análisis al archivo *views.py*

```
views.py:26:0: W0614: Unused import View from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import settings from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import base64 from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import safe_string from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import exceptions from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import ContentFile from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import File from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import strip_tags from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import html_safe from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import basestring from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import serializers from wildcard import (unused-wildcard-import)
views.py:28:0: C0411: third party import "from notifications.signals import notify" should be placed before "from BookingQBA.settings import STATICFILES_DIRS, BASE_DIR" (wrong-import-order)
views.py:29:0: C0411: third party import "from notifications import models as models_notify" should be placed before "from BookingQBA.settings import STATICFILES_DIRS, BASE_DIR" (wrong-import-order)
views.py:33:0: C0411: third party import "import sweetify" should be placed before "from BookingQBA.settings import STATICFILES_DIRS, BASE_DIR" (wrong-import-order)
views.py:20:0: C0412: Imports from package django are not grouped (ungrouped-imports)
views.py:31:0: C0412: Imports from package BookingQBAapp are not grouped (ungrouped-imports)

-----
Your code has been rated at 5.67/10
```

Ilustración 36 Resultado del análisis estático del código

5.4 Análisis de costos y beneficios.

Para el análisis de costos y beneficios se ha tenido en cuenta la cantidad de horas dedicadas al sistema en las distintas etapas por las que transita. Se establece para este proyecto 8 horas de trabajo diarias, en 24 días laborables por mes. Por tanto, el total de horas trabajadas en un mes será de 192 horas. El salario del desarrollador del proyecto es \$ 700.00 CUP.

Tabla 13 Esfuerzo del implementador del sistema

Tareas desarrolladas	Cantidad de trabajadores	Salario por hora	Tiempo en horas consumidas	Total
Captura de requisitos	1	3.65	30 h (8*5 días)	\$ 109.50
Diseño de la solución propuesta	1	3.65	192 h (8*24 días)	\$ 700.80
Selección y estudio de herramientas	1	3.65	192 h (8*24 días)	\$ 700.80
Implementación y documentación	1	3.65	1920 h (8*240 días)	\$ 7008.00
Implantación y capacitación	1	3.65	40 h (8*5 días)	\$ 146.00
Pruebas	1	3.65	40 h (8*5 días)	\$ 146.00
TOTAL			2414 h	\$ 8811.10

Para un total de 2414 horas y \$ 8811.10 CUP y por concepto de consumo de electricidad, medios de cómputo, gasto de alimento, papel y otros, se estima un gasto adicional de 5 pesos por hora. Se estima que aproximadamente el tiempo de desarrollo del proyecto es de un año y un mes para un costo total de \$ 20881,10 CUP. Para la implantación del sistema será necesario hospedarlo en ETECSA, lo cual aumenta el costo pos desarrollo en \$ 20.00 CUC/mes[33]. La

misma aplicación desarrollada por una compañía productora donde el servicio por hora de desarrollo de un software es de \$10, ascendería el costo a \$ 24140.00 CUP.

5.5 Beneficios tangibles e intangibles.

Beneficios tangibles

- Se evita invertir en la compra de software y licencia del mismo para la gestión de los procesos, según los cálculos costaría \$ 20881,10 CUP.
- Se logra obtener un sistema propio, cuya administración de la información será de manera centralizada.
- Informatización de las estadísticas de los hospedajes.

Beneficios intangibles

- El usuario puede consultar información actualizada en cualquier momento que requiera de la misma.
- Los reportes de análisis estadísticos obtenidos, ayudan a la toma de decisiones
- Se reduce el tiempo de búsqueda de información de los hospedajes en Cuba.
- Procesamiento de la información de forma ágil.

5.6 Conclusiones.

- a) Se ejecutaron pruebas automatizadas, para resolver o evitar problemas en el desarrollo de la aplicación.
- b) Se identificaron los beneficios tangibles e intangibles que posee la aplicación implementada respondiendo los mismos al valor práctico esperado por el cliente.
- c) Se realizó un estudio de factibilidad del sistema propuesto, cuyo costo asciende a \$ 20881.10 CUP, desarrollado durante trece meses. Por lo que se considera al sistema como una propuesta factible, acorde a los parámetros acordados con el cliente.

Conclusiones

1. A partir del análisis de la bibliografía, se propone desarrollar una aplicación que ofrezca de manera integral mayores beneficios a la gestión del proceso de hospedaje en Cuba, utilizando las tecnologías acordes para un óptimo resultado.
2. Se diseñó el modelo de dominio y se establecieron las reglas del negocio para identificar las necesidades, tanto de los arrendatarios como de los usuarios.
3. Se identificaron los requisitos funcionales y no funcionales agrupados por paquetes o conjuntos, respectivamente; visualizando y describiendo los casos de uso del sistema para analizar y diseñar la propuesta de solución.
4. Fueron aplicados los principios y patrones de diseño acordes para la implementación de la Aplicación Web, así como el diseño de la estructura de la base de datos.
5. Las pruebas al sistema permitieron verificar el buen funcionamiento del mismo, obteniéndose los resultados esperados, mejorando la calidad del software, reduciendo el número de errores.
6. El estudio de factibilidad del sistema propuesto arrojó un costo de \$ 20881.10 CUP, en los tiempos definidos con el cliente, determinado lo factible del desarrollo.

Recomendaciones

Se enumeran a continuación lo que se considera como las líneas de trabajo futuras inmediatas, una vez hospedada la aplicación:

- Integrar la aplicación con *Google Analytics* (Herramienta gratuita de analítica web) para obtener resultados estadísticos.
- Aplicar un análisis SEO (por sus siglas en inglés *Search Engine Optimization* que se traduce, Optimización para motores de búsqueda) para mejorar el posicionamiento del sitio web en los motores de búsqueda como Google, Yahoo, Bing, etc.

Referencias bibliográficas

- [1] Ministerio de Justicia, «Gaceta Oficial de la República de Cuba». 15-jul-2014.
- [2] «Cierre estadístico mensual de la Empresa de Trabajo Provincial». nov-2018.
- [3] «Informatización de la Sociedad - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Informatizaci%C3%B3n_de_la_Sociedad. [Accedido: 07-ene-2019].
- [4] «RealCasaRenta - Alojamientos en Cuba de Casas Particulares». [En línea]. Disponible en: <https://www.realcasarenta.com/>. [Accedido: 07-ene-2019].
- [5] «Cuba 2019! Casas particulares Cuba, alquiler apartamentos La Habana, alojamientos y casas familia , habitaciones renta». [En línea]. Disponible en: <http://www.casas-cuba.org/index.html>. [Accedido: 07-ene-2019].
- [6] «HospedajeCubano.com - Renta de Habitaciones y Casas Particulares en Cuba». [En línea]. Disponible en: <http://hospedajecubano.com/>. [Accedido: 07-ene-2019].
- [7] «PORTADA | RENT.CU». [En línea]. Disponible en: <http://rent.cu/>. [Accedido: 07-ene-2019].
- [8] «¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML.» [En línea]. Disponible en: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163. [Accedido: 07-ene-2019].
- [9] «Qué es SCRUM», *Proyectos Ágiles*, 04-ago-2008. .
- [10] «Metodología RUP», *Metodoss*, 19-may-2016. .
- [11] «Metodologia Rup». .
- [12] «Características de Enterprise Architect herramienta de modelado UML». [En línea]. Disponible en: <http://www.sparxsystems.com.ar/products/ea/features.html>. [Accedido: 07-ene-2019].
- [13] «Guión Visual Paradigm for UML», *Tutorial Visual Pradigm*. [En línea]. Disponible en: <http://www.ie.inf.uc3m.es/grupo/docencia/reglada/ls1y2/PracticaVP.pdf>. [Accedido: 07-ene-2019].
- [14] «Aplicaciones Web Vs Escritorio - INTERNET YA». [En línea]. Disponible en: <https://www.internetya.co/aplicaciones-web-vs-escritorio/>. [Accedido: 07-ene-2019].
- [15] «Conceptos básicos del lenguaje Java», 03-dic-2012. [En línea]. Disponible en: <http://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/index.html>. [Accedido: 07-ene-2019].
- [16] DesarrolloWeb.com, «Qué es Python», *DesarrolloWeb.com*. [En línea]. Disponible en: <http://www.desarrolloweb.com/articulos/1325.php>. [Accedido: 07-ene-2019].
- [17] «PHP: ¿Qué es PHP? - Manual». [En línea]. Disponible en: <http://php.net/manual/es/intro-what-is.php>. [Accedido: 07-ene-2019].
- [18] «Sistemas de bases de datos. Informatica Aplicada a la Gestion Publica. Universidad de Murcia. Rafael Barzanallana.» [En línea]. Disponible en: http://dis.um.es/~barzana/Informatica/IAGP/IAGP_Sistemas_BD.html. [Accedido: 03-jun-2019].
- [19] «SQLite: La Base de Datos Embebida», *SG Buzz*. [En línea]. Disponible en: <https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida>. [Accedido: 07-ene-2019].
- [20] «Qué es PostgreSQL», *OpenWebinars.net*. [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-postgresql/>. [Accedido: 07-ene-2019].

- [21] «¿Qué es MySQL?», *tuprogramacion.com*. [En línea]. Disponible en: <http://www.tuprogramacion.com/glosario/que-es-mysql/>. [Accedido: 07-ene-2019].
- [22] Edgar, «5 razones por las cuales debes usar PostgreSQL», *Guiadev*, 26-sep-2015. [En línea]. Disponible en: <https://guiadev.com/5-razones-por-las-cuales-debes-usar-postgresql/>. [Accedido: 03-jun-2019].
- [23] «¿Qué es Django? · Django Girls Tutorial». [En línea]. Disponible en: <https://tutorial.djangogirls.org/es/django/>. [Accedido: 07-ene-2019].
- [24] «Qué es Django y por qué usarlo», *OpenWebinars.net*, 03-ago-2018. [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>. [Accedido: 07-ene-2019].
- [25] «Turbogears Framework para el desarrollo de aplicaciones web», *Scribd*. [En línea]. Disponible en: <https://es.scribd.com/document/15021228/Turbogears-Framework-para-el-desarrollo-de-aplicaciones-web>. [Accedido: 07-ene-2019].
- [26] andrearrrs, «Flask: minimalismo para el desarrollo web en Python», *Hipertextual*, 13-ago-2014. [En línea]. Disponible en: <https://hipertextual.com/archivo/2014/08/flask-python/>. [Accedido: 07-ene-2019].
- [27] «¿Qué es Flask?», *OpenWebinars.net*, 17-nov-2017. [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-flask/>. [Accedido: 07-ene-2019].
- [28] J. G. D. en informática, P. en la C. M. I. en I. P. vez que me tocó hacer una gestión Á. en una empresa año 2001 D. entonces he trabajado en, O. Para, y más de 90 Y. he formado a más de 2000 alumnos T. soy profe de la U. R. J. Carlos, «Los patrones GRASP», *Javier Garzás*, 05-ago-2014. .
- [29] M. G. Cuesta, «Patrones de diseño - Agrupación GoF - Gang of Four | Aprende Java en Español», *Patrones de diseño - Agrupación GoF - Gang of Four | Aprende Java en Español*, 06-abr-2016. .
- [30] «Testing de modelos en Django, buenas prácticas - Software Crafters». [En línea]. Disponible en: <https://softwarecrafters.io/django/testing-modelos>. [Accedido: 30-may-2019].
- [31] «Software QA - ¿Cuáles son los tipos de pruebas software?», *Panel Sistemas*, 11-feb-2015. .
- [32] M. Mazzarri, «Pylint: Análisis estático del código en Python», *milmazz*, 12-mar-2010. [En línea]. Disponible en: <https://milmazz.uno/article/2010/03/12/pylint-analisis-estatico-del-codigo-en-python/>. [Accedido: 04-jun-2019].
- [33] «Etecsa ofrece servicio de hospedaje de sitios webs a personas naturales | Cubadebate». [En línea]. Disponible en: <http://www.cubadebate.cu/noticias/2019/04/29/etecsa-ofrece-servicio-de-hospedaje-de-sitios-webs-a-personas-naturales/>. [Accedido: 29-may-2019].

Bibliografía

1. Ministerio de Justicia, «Gaceta Oficial de la República de Cuba». 15-jul-2014.
2. «Cierre estadístico mensual de la Empresa de Trabajo Provincial». nov-2018.
3. «Informatización de la Sociedad - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Informatizaci%C3%B3n_de_la_Sociedad. [Accedido: 07-ene-2019].
4. «RealCasaRenta - Alojamientos en Cuba de Casas Particulares». [En línea]. Disponible en: <https://www.realcasarenta.com/>. [Accedido: 07-ene-2019].
5. «Cuba 2019! Casas particulares Cuba, alquiler apartamentos La Habana, alojamientos y casas familia , habitaciones renta». [En línea]. Disponible en: <http://www.casas-cuba.org/index.html>. [Accedido: 07-ene-2019].
6. «HospedajeCubano.com - Renta de Habitaciones y Casas Particulares en Cuba». [En línea]. Disponible en: <http://hospedajecubano.com/>. [Accedido: 07-ene-2019].
7. «PORTADA | RENT.CU». [En línea]. Disponible en: <http://rent.cu/>. [Accedido: 07-ene-2019].
8. «¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML.» [En línea]. Disponible en: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163. [Accedido: 07-ene-2019].
9. «Qué es SCRUM», Proyectos Ágiles, 04-ago-2008. .
10. «Metodología RUP», Metodoss, 19-may-2016. .
11. «Metodologia Rup». .
12. «Características de Enterprise Architect herramienta de modelado UML». [En línea]. Disponible en: <http://www.sparxsystems.com.ar/products/ea/features.html>. [Accedido: 07-ene-2019].
13. «Guión Visual Paradigm for UML», Tutorial Visual Pradigm. [En línea]. Disponible en: <http://www.ie.inf.uc3m.es/grupo/docencia/reglada/ls1y2/PracticaVP.pdf>. [Accedido: 07-ene-2019].

14. «Aplicaciones Web Vs Escritorio - INTERNET YA». [En línea]. Disponible en: <https://www.internetya.co/aplicaciones-web-vs-escritorio/>. [Accedido: 07-ene-2019].
15. «Conceptos básicos del lenguaje Java», 03-dic-2012. [En línea]. Disponible en: <http://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/index.html>. [Accedido: 07-ene-2019].
16. DesarrolloWeb.com, «Qué es Python», DesarrolloWeb.com. [En línea]. Disponible en: <http://www.desarrolloweb.com/articulos/1325.php>. [Accedido: 07-ene-2019].
17. «PHP: ¿Qué es PHP? - Manual». [En línea]. Disponible en: <http://php.net/manual/es/intro-what-is.php>. [Accedido: 07-ene-2019].
18. «Sistemas de bases de datos. Informatica Aplicada a la Gestion Publica. Universidad de Murcia. Rafael Barzanallana.» [En línea]. Disponible en: http://dis.um.es/~barzana/Informatica/IAGP/IAGP_Sistemas_BD.html. [Accedido: 03-jun-2019].
19. «SQLite: La Base de Datos Embebida», SG Buzz. [En línea]. Disponible en: <https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida>. [Accedido: 07-ene-2019].
20. «Qué es PostgreSQL», OpenWebinars.net. [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-postgresql/>. [Accedido: 07-ene-2019].
21. «¿Qué es MySQL?», tuprogramacion.com. [En línea]. Disponible en: <http://www.tuprogramacion.com/glosario/que-es-mysql/>. [Accedido: 07-ene-2019].
22. Edgar, «5 razones por las cuales debes usar PostgreSQL», Guiadev, 26-sep-2015. [En línea]. Disponible en: <https://guiadev.com/5-razones-por-las-cuales-debes-usar-postgresql/>. [Accedido: 03-jun-2019].
23. «¿Qué es Django? • Django Girls Tutorial». [En línea]. Disponible en: <https://tutorial.djangogirls.org/es/django/>. [Accedido: 07-ene-2019].
24. «Qué es Django y por qué usarlo», OpenWebinars.net, 03-ago-2018. [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>. [Accedido: 07-ene-2019].
25. «Turbogears Framework para el desarrollo de aplicaciones web», Scribd. [En línea]. Disponible en: <https://es.scribd.com/document/15021228/Turbogears-Framework-para-el-desarrollo-de-aplicaciones-web>. [Accedido: 07-ene-2019].

26. andrearrs, «Flask: minimalismo para el desarrollo web en Python», Hipertextual, 13-ago-2014. [En línea]. Disponible en: <https://hipertextual.com/archivo/2014/08/flask-python/>. [Accedido: 07-ene-2019].
27. «¿Qué es Flask?», OpenWebinars.net, 17-nov-2017. [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-flask/>. [Accedido: 07-ene-2019].
28. J. G. D. en informática, P. en la C. M. I. en I. P. vez que me tocó hacer una gestión Á. en una empresa año 2001 D. entonces he trabajado en, O. Para, y más de 90 Y. he formado a más de 2000 alumnos T. soy profe de la U. R. J. Carlos, «Los patrones GRASP», Javier Garzás, 05-ago-2014. .
29. M. G. Cuesta, «Patrones de diseño - Agrupación GoF - Gang of Four | Aprende Java en Español», Patrones de diseño - Agrupación GoF - Gang of Four | Aprende Java en Español, 06-abr-2016. .
30. «Testing de modelos en Django, buenas prácticas - Software Crafters». [En línea]. Disponible en: <https://softwarecrafters.io/django/testing-modelos>. [Accedido: 30-may-2019].
31. «Software QA - ¿Cuáles son los tipos de pruebas software?», Panel Sistemas, 11-feb-2015. .
32. M. Mazzarri, «Pylint: Análisis estático del código en Python», milmazz, 12-mar-2010. [En línea]. Disponible en: <https://milmazz.uno/article/2010/03/12/pylint-analisis-estatico-del-codigo-en-python/>. [Accedido: 04-jun-2019].
33. «Etecsa ofrece servicio de hospedaje de sitios webs a personas naturales | Cubadebate». [En línea]. Disponible en: <http://www.cubadebate.cu/noticias/2019/04/29/etecsa-ofrece-servicio-de-hospedaje-de-sitios-webs-a-personas-naturales/>. [Accedido: 29-may-2019].
34. «25.3. unittest — Unit testing framework — Python 2.7.16rc1 documentation». [En línea]. Disponible en: <https://docs.python.org/2/library/unittest.html>. [Accedido: 26-feb-2019].
35. «Patrones Gof - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Patrones_Gof. [Accedido: 31-may-2019].
36. «Qué es PostgreSQL | OpenWebinars». [En línea]. Disponible en: <https://openwebinars.net/blog/que-es-postgresql/>. [Accedido: 07-ene-2019].

Glosario de siglas y términos

API: Es un conjunto de funciones y procedimientos (o métodos, en la programación orientado a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

APK: Formato de archivo que se usa para empaquetar aplicaciones para el sistema operativo Android. Se hace referencia en el documento a la aplicación para teléfonos inteligentes.

Mapeo relacional de objetos (ORM por sus siglas en inglés): Es una técnica de programación para convertir datos entre el sistema de tipos utilizando un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

Marco de trabajo (*frameworks*): Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas; para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

PC: Computadora personal.

Red Cuba: Intranet con alcance nacional.

Responsive: Adaptable a cualquier tipo de dispositivo, dígase móvil, tableta o laptop.

REST: Interfaz entre sistemas que utiliza directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos en diferentes formatos (XML, JSON, etc.)

Software: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

WIFI: Tecnología que permite interconexión inalámbrica de dispositivos electrónicos.

Anexo 1: Descripción de casos de uso del sistema.

Tabla 14 Descripción del caso de uso <Cambiar Contraseña>

Nombre del caso de uso	Cambiar Contraseña
Actores	Usuario
Resumen	Se inicia cuando el usuario quiere cambiar la contraseña. La aplicación verifica si la contraseña antigua es correcta y en caso de ser incorrecta, se muestra un mensaje de error, así también en caso de que la contraseña nueva no coincida con la confirmación de la misma. El caso de uso termina cuando el sistema le permite cambiar o no la contraseña.
Precondiciones	El usuario debe estar autenticado en el sistema.
Poscondiciones	El usuario tiene una nueva contraseña.

Tabla 15 Descripción del caso de uso <Salvar base de datos>

Nombre del caso de uso	Salvar base de datos
Actores	Administrador
Resumen	Se inicia cuando el usuario quiere salvar la base de datos. La aplicación verifica si existe conexión con la base de datos y ejecuta el comando de salva. En caso de error notifica al usuario, así como si fue satisfactorio. El caso de uso termina con la notificación.
Precondiciones	El administrador debe estar autenticado en el sistema.
Poscondiciones	Se crea el fichero con formato sql con la salva de la información.

Tabla 16 Descripción del caso de uso <Restaurar base de datos>

Nombre del caso de uso	Restaurar base de datos
Actores	Administrador
Resumen	Se inicia cuando el usuario quiere restaurar la base de datos. La aplicación verifica si existe conexión con la base de datos y ejecuta el comando de restaurar. En caso de error notifica al usuario, así como si fue satisfactorio. El caso de uso termina con la notificación.
Precondiciones	El administrador debe estar autenticado en el sistema.
Poscondiciones	Se actualiza la base de datos.

Tabla 17 Descripción del caso de uso <Visualizar trazas>

Nombre del caso de uso	Visualizar trazas
Actores	Administrador
Resumen	Se inicia cuando el usuario quiere visualizar las trazas del sistema. El caso de uso termina mostrando la tabla con todos los registros, dando la posibilidad de filtrar y de buscar en la misma.
Precondiciones	El administrador debe estar autenticado en el sistema.

Tabla 18 Descripción del caso de uso <Administrar roles>

Nombre del caso de uso	Administrar roles
Actores	Administrador
Resumen	El caso de uso se inicia cuando el Administrador desea insertar, modificar un rol. El sistema muestra un listado de los roles registrados mostrando las opciones de insertar y modificar. El caso de uso termina cuando se realiza una de las opciones.
Precondiciones	El administrador debe estar autenticado en el sistema.
Poscondiciones	Se actualiza la base de datos.

Tabla 19 Descripción del caso de uso <Gestionar nomencladores>

Nombre del caso de uso	Gestionar nomencladores
Actores	WebMaster
Resumen	<p>El actor puede agregar, modificar y eliminar:</p> <ul style="list-style-type: none"> - Facilidades - Tipo de monedas - Tipo de punto de interés - Punto de interés - Zona de referencia <p>Al visualizar el listado las acciones de gestión están facilitadas en cada registro con botones representativos.</p>
Precondiciones	El administrador debe estar autenticado en el sistema.
Poscondiciones	Se actualiza la base de datos.

Tabla 20 Descripción del caso de uso <Gestionar hospedaje>

Nombre del caso de uso	Gestionar hospedaje
Actores	Arrendatario, WebMaster
Resumen	El actor puede agregar, modificar y desactivar su hospedaje. El sistema valida los datos y emite una notificación de error o no. El caso de uso termina cuando se registra el hospedaje.
Precondiciones	El actor debe estar autenticado en el sistema.
Poscondiciones	Se actualiza la base de datos. El usuario puede agregar fotos de la casa.

Tabla 21 Descripción del caso de uso <Gestionar oferta>

Nombre del caso de uso	Gestionar oferta
Actores	Arrendatario, WebMaster
Resumen	El actor puede agregar, modificar y eliminar la oferta. El sistema valida los datos y emite una notificación de error o no. El caso de uso termina cuando se registra la oferta.
Precondiciones	El actor debe estar autenticado en el sistema.
Poscondiciones	Se actualiza la base de datos.

Tabla 22 Descripción del caso de uso <Visualizar reportes>

Nombre del caso de uso	Visualizar reportes
Actores	WebMaster
Resumen	El actor puede visualizar los reportes una vez que se encuentre en el sistema. El sistema muestra gráficos referentes a los datos mostrados en la tabla de registro. Se pueden exportar a PDF los resultados.
Precondiciones	El actor debe estar autenticado en el sistema.
Poscondiciones	Se genera el fichero con formato PDF.

Tabla 23 Descripción del caso de uso <Tasa de cambio>

Nombre del caso de uso	Tasa de cambio
Actores	WebMaster
Resumen	El caso de uso comienza cuando el WebMaster se encuentra en la interfaz de gestión de la tasa de cambio. Puede seleccionar el día para la inserción o modificación de los datos. En caso de dejar en blanco USD, CUC y CUP, se guardan datos por defecto. El sistema valida los datos que sean solo números. Notifica al usuario. El caso de uso termina cuando se guardan los datos.
Precondiciones	El actor debe estar autenticado en el sistema.
Poscondiciones	Se guardan los datos en la base de datos.

Anexo 2: Pruebas automáticas a la Aplicación.

```
E:\escuela\Tesis\BookingQBA>python manage.py test -v 2
```

```
Creating test database for alias 'default' ('test_bookingqbadb1')...
```

```
Operations to perform:
```

```
  Synchronize unmigrated apps: messages, rest_framework, staticfiles, sweetify
```

```
  Apply all migrations: BookingQBAapp, admin, auth, authtoken, captcha, contenttypes, notifications, sessions
```

```
Synchronizing apps without migrations:
```

```
  Creating tables...
```

```
    Running deferred SQL...
```

```
Running migrations:
```

```
  Applying contenttypes.0001_initial... OK
```

```
  Applying contenttypes.0002_remove_content_type_name... OK
```

```
  Applying auth.0001_initial... OK
```

```
  Applying auth.0002_alter_permission_name_max_length... OK
```

```
  Applying auth.0003_alter_user_email_max_length... OK
```

```
  Applying auth.0004_alter_user_username_opts... OK
```

```
  Applying auth.0005_alter_user_last_login_null... OK
```

```
  Applying auth.0006_require_contenttypes_0002... OK
```

```
  Applying auth.0007_alter_validators_add_error_messages... OK
```

```
  Applying auth.0008_alter_user_username_max_length... OK
```

```
  Applying BookingQBAapp.0001_initial... OK
```

```
  Applying BookingQBAapp.0002_auto_20190319_1118... OK
```

```
  Applying BookingQBAapp.0003_auto_20190319_1143... OK
```

```
  Applying BookingQBAapp.0004_userapp... OK
```

```
  Applying BookingQBAapp.0005_auto_20190323_0357... OK
```

```
  Applying BookingQBAapp.0006_auto_20190323_0413... OK
```

```
  Applying BookingQBAapp.0007_auto_20190323_0425... OK
```

```
  Applying BookingQBAapp.0008_auto_20190323_1936... OK
```

```
  Applying BookingQBAapp.0009_auto_20190406_0007... OK
```

```
  Applying BookingQBAapp.0010_auto_20190408_0949... OK
```

```
  Applying BookingQBAapp.0011_auto_20190408_1008... OK
```

```
  Applying BookingQBAapp.0012_subscriptionmessage_subscription... OK
```

```
  Applying BookingQBAapp.0013_auto_20190414_2348... OK
```

```
  Applying BookingQBAapp.0014_auto_20190415_1253... OK
```

```
  Applying BookingQBAapp.0015_userdevice... OK
```

```
  Applying BookingQBAapp.0016_auto_20190419_0102... OK
```

```
  Applying BookingQBAapp.0017_auto_20190419_0920... OK
```

```
  Applying BookingQBAapp.0018_auto_20190419_1322... OK
```

```
  Applying BookingQBAapp.0019_auto_20190419_1350... OK
```

Applying BookingQBAapp.0020_auto_20190419_1416... OK
Applying BookingQBAapp.0021_auto_20190419_2246... OK
Applying BookingQBAapp.0022_rentrating_description... OK
Applying BookingQBAapp.0023_auto_20190427_0014... OK
Applying BookingQBAapp.0024_poitype_image... OK
Applying BookingQBAapp.0025_remove_rent_slogan... OK
Applying BookingQBAapp.0026_auto_20190429_1153... OK
Applying BookingQBAapp.0027_auto_20190429_1154... OK
Applying BookingQBAapp.0028_auto_20190429_1159... OK
Applying BookingQBAapp.0029_auto_20190429_1256... OK
Applying BookingQBAapp.0030_rentcomment_uui... OK
Applying BookingQBAapp.0031_auto_20190506_1109... OK
Applying BookingQBAapp.0032_rentcomment_emotion... OK
Applying BookingQBAapp.0033_auto_20190518_2330... OK
Applying BookingQBAapp.0034_auto_20190518_2330... OK
Applying BookingQBAapp.0035_auto_20190518_2333... OK
Applying BookingQBAapp.0036_auto_20190518_2336... OK
Applying BookingQBAapp.0037_auto_20190520_1151... OK
Applying BookingQBAapp.0038_poi_active... OK
Applying BookingQBAapp.0039_remove_poi_active... OK
Applying BookingQBAapp.0040_auto_20190529_1304... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying authtoken.0001_initial... OK
Applying authtoken.0002_auto_20160226_1747... OK
Applying captcha.0001_initial... OK
Applying notifications.0001_initial... OK
Applying notifications.0002_auto_20150224_1134... OK
Applying notifications.0003_notification_data... OK
Applying notifications.0004_auto_20150826_1508... OK
Applying notifications.0005_auto_20160504_1520... OK
Applying notifications.0006_indexes... OK
Applying sessions.0001_initial... OK
System check identified no issues (0 silenced).
test_rent_creation_100 (BookingQBAapp.tests.Rent100CreateTestModel) ... ok
test_rent_creation_100 (BookingQBAapp.tests.RentCreateTestModel) ... ERROR

=====

ERROR: test_rent_creation_100 (BookingQBAapp.tests.RentCreateTestModel)

Traceback (most recent call last):

File "E:\escuela\Tesis\BookingQBA\BookingQBAapp\tests.py", line 18, in setUp

```
capability=5, maxBeds=2, maxBath=2, maxRooms=3)
File "C:\Program Files\Python37\lib\site-packages\django-1.11.20-
py3.7.egg\django\db\models\manager.py", line 85, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
File "C:\Program Files\Python37\lib\site-packages\django-1.11.20-py3.7.egg\django\db\models\query.py",
line 392, in create
    obj = self.model(**kwargs)
File "C:\Program Files\Python37\lib\site-packages\django-1.11.20-py3.7.egg\django\db\models\base.py",
line 573, in __init__
    raise TypeError("%s' is an invalid keyword argument for this function" % list(kwargs)[0])
TypeError: 'slogan' is an invalid keyword argument for this function
```

Ran 2 tests in 1.789s

FAILED (errors=1)

Destroying test database for alias 'default' ('test_bookingbadb1')...

Anexo 3: Análisis Estático al código del fichero *views.py*.

E:\escuela\Tesis\BookingQBA\BookingQBAApp>pylint views.py

***** Module BookingQBAApp.views

views.py:43:0: C0301: Line too long (103/100) (line-too-long)

views.py:58:0: C0301: Line too long (115/100) (line-too-long)

views.py:80:0: C0301: Line too long (120/100) (line-too-long)

views.py:87:0: C0301: Line too long (102/100) (line-too-long)

views.py:89:0: C0301: Line too long (104/100) (line-too-long)

views.py:91:0: C0301: Line too long (106/100) (line-too-long)

views.py:105:0: C0301: Line too long (109/100) (line-too-long)

views.py:125:0: C0301: Line too long (110/100) (line-too-long)

views.py:325:30: C0326: Exactly one space required after comma

sweetify.error(request,msgError)

^ (bad-whitespace)

views.py:352:0: C0301: Line too long (103/100) (line-too-long)

views.py:353:0: C0301: Line too long (112/100) (line-too-long)

views.py:354:0: C0301: Line too long (114/100) (line-too-long)

views.py:357:0: C0301: Line too long (102/100) (line-too-long)

views.py:372:0: C0301: Line too long (111/100) (line-too-long)

views.py:373:0: C0301: Line too long (120/100) (line-too-long)

views.py:378:0: C0301: Line too long (110/100) (line-too-long)

views.py:379:0: C0301: Line too long (104/100) (line-too-long)

views.py:380:0: C0301: Line too long (102/100) (line-too-long)

views.py:381:0: C0301: Line too long (101/100) (line-too-long)

views.py:389:0: C0301: Line too long (107/100) (line-too-long)

views.py:437:0: C0301: Line too long (106/100) (line-too-long)

views.py:440:0: C0301: Line too long (109/100) (line-too-long)

views.py:464:0: C0301: Line too long (119/100) (line-too-long)

views.py:473:0: C0301: Line too long (109/100) (line-too-long)

views.py:477:0: C0301: Line too long (105/100) (line-too-long)

views.py:490:0: C0301: Line too long (104/100) (line-too-long)

views.py:497:0: C0301: Line too long (109/100) (line-too-long)

views.py:498:0: C0301: Line too long (134/100) (line-too-long)

views.py:499:0: C0301: Line too long (154/100) (line-too-long)

views.py:500:0: C0330: Wrong hanging indentation (add 116 spaces).

msg.startDate) + "\n" \

^

| (bad-continuation)

views.py:502:0: C0301: Line too long (110/100) (line-too-long)

views.py:503:0: C0301: Line too long (191/100) (line-too-long)

views.py:504:0: C0301: Line too long (155/100) (line-too-long)

views.py:505:0: C0301: Line too long (135/100) (line-too-long)

views.py:507:0: C0301: Line too long (102/100) (line-too-long)

views.py:510:0: C0301: Line too long (106/100) (line-too-long)

views.py:528:0: C0301: Line too long (108/100) (line-too-long)

views.py:531:0: C0301: Line too long (115/100) (line-too-long)

views.py:552:0: C0301: Line too long (113/100) (line-too-long)

views.py:553:0: C0301: Line too long (130/100) (line-too-long)

views.py:554:0: C0301: Line too long (106/100) (line-too-long)

views.py:555:0: C0301: Line too long (136/100) (line-too-long)

views.py:605:0: C0301: Line too long (104/100) (line-too-long)

views.py:610:0: C0301: Line too long (118/100) (line-too-long)

views.py:613:0: C0301: Line too long (116/100) (line-too-long)

views.py:616:0: C0301: Line too long (117/100) (line-too-long)

views.py:617:0: C0301: Line too long (105/100) (line-too-long)

views.py:651:0: C0301: Line too long (116/100) (line-too-long)

views.py:672:0: C0301: Line too long (107/100) (line-too-long)

views.py:694:0: C0301: Line too long (137/100) (line-too-long)

views.py:694:89: C0326: Exactly one space required after comma

```

        return                                render(request,                                "Admin/backend.html",
{'adminViewsGraph':adminViewsGraph(),'graph_views_count': admin_graph_rent_visit()})
                                ^ (bad-whitespace)

```

views.py:719:0: C0301: Line too long (102/100) (line-too-long)

views.py:767:0: C0301: Line too long (107/100) (line-too-long)

views.py:787:0: C0301: Line too long (107/100) (line-too-long)

views.py:806:0: C0301: Line too long (115/100) (line-too-long)

views.py:833:0: C0301: Line too long (115/100) (line-too-long)

views.py:851:0: C0301: Line too long (116/100) (line-too-long)

views.py:1042:0: C0301: Line too long (103/100) (line-too-long)

views.py:1070:0: C0301: Line too long (137/100) (line-too-long)

views.py:1070:84: C0326: Exactly one space required after comma

```

        return render(request, 'Admin/Rent/rent.html', {'municipalities': municipalities,'rents': rents, 'form': form,
'userForm': userForm})
                                ^ (bad-whitespace)

```

views.py:1137:0: C0301: Line too long (107/100) (line-too-long)

views.py:1138:0: C0301: Line too long (119/100) (line-too-long)

views.py:1140:0: C0301: Line too long (104/100) (line-too-long)

views.py:1142:0: C0330: Wrong continued indentation (remove 5 spaces).

```

        'a petición del WebMaster ' + request.user.username, level='warning')
    | ^ (bad-continuation)

```

views.py:1164:0: C0301: Line too long (101/100) (line-too-long)

views.py:1165:0: C0301: Line too long (126/100) (line-too-long)

views.py:1166:0: C0301: Line too long (146/100) (line-too-long)

views.py:1167:0: C0330: Wrong hanging indentation (add 116 spaces).

```
msg.startDate) + "\n" \
```

```
^
```

| (bad-continuation)

views.py:1169:0: C0301: Line too long (102/100) (line-too-long)

views.py:1170:0: C0301: Line too long (183/100) (line-too-long)

views.py:1171:0: C0301: Line too long (147/100) (line-too-long)

views.py:1172:0: C0301: Line too long (127/100) (line-too-long)

views.py:1192:0: C0301: Line too long (101/100) (line-too-long)

views.py:1199:0: C0301: Line too long (101/100) (line-too-long)

views.py:1210:0: C0301: Line too long (124/100) (line-too-long)

views.py:1311:0: C0301: Line too long (104/100) (line-too-long)

views.py:1341:0: C0301: Line too long (103/100) (line-too-long)

views.py:1343:0: C0301: Line too long (118/100) (line-too-long)

views.py:1347:0: C0301: Line too long (118/100) (line-too-long)

views.py:1351:0: C0301: Line too long (118/100) (line-too-long)

views.py:1353:0: C0301: Line too long (111/100) (line-too-long)

views.py:1355:0: C0301: Line too long (118/100) (line-too-long)

views.py:1374:0: C0301: Line too long (110/100) (line-too-long)

views.py:1386:0: C0301: Line too long (107/100) (line-too-long)

views.py:1409:0: C0301: Line too long (102/100) (line-too-long)

views.py:1423:0: C0301: Line too long (103/100) (line-too-long)

views.py:1428:0: C0301: Line too long (102/100) (line-too-long)

views.py:1445:0: C0301: Line too long (107/100) (line-too-long)

views.py:1447:0: C0301: Line too long (101/100) (line-too-long)

views.py:1449:0: C0301: Line too long (107/100) (line-too-long)

views.py:1451:0: C0301: Line too long (113/100) (line-too-long)

views.py:1460:0: C0301: Line too long (103/100) (line-too-long)

views.py:1467:0: C0301: Line too long (119/100) (line-too-long)

views.py:1627:0: C0301: Line too long (116/100) (line-too-long)

views.py:1644:0: C0301: Line too long (232/100) (line-too-long)

views.py:1648:0: C0301: Line too long (136/100) (line-too-long)

views.py:1664:0: C0301: Line too long (105/100) (line-too-long)

views.py:1673:0: C0301: Line too long (185/100) (line-too-long)

views.py:1694:0: C0301: Line too long (113/100) (line-too-long)

views.py:1696:0: C0301: Line too long (101/100) (line-too-long)

views.py:1701:0: C0301: Line too long (109/100) (line-too-long)

views.py:1702:0: C0301: Line too long (118/100) (line-too-long)

views.py:1707:0: C0301: Line too long (105/100) (line-too-long)

views.py:1708:0: C0301: Line too long (114/100) (line-too-long)

views.py:1715:0: C0301: Line too long (101/100) (line-too-long)

views.py:1716:0: C0301: Line too long (110/100) (line-too-long)

views.py:1767:0: C0301: Line too long (112/100) (line-too-long)

views.py:1790:0: C0301: Line too long (116/100) (line-too-long)
views.py:1812:0: C0301: Line too long (112/100) (line-too-long)
views.py:1817:0: C0301: Line too long (101/100) (line-too-long)
views.py:1833:0: C0301: Line too long (112/100) (line-too-long)
views.py:1854:0: C0301: Line too long (112/100) (line-too-long)
views.py:1875:0: C0301: Line too long (107/100) (line-too-long)
views.py:1896:0: C0301: Line too long (113/100) (line-too-long)
views.py:1898:0: C0301: Line too long (107/100) (line-too-long)
views.py:1902:0: C0301: Line too long (115/100) (line-too-long)
views.py:1919:0: C0301: Line too long (112/100) (line-too-long)
views.py:1940:0: C0301: Line too long (117/100) (line-too-long)
views.py:1962:0: C0301: Line too long (109/100) (line-too-long)
views.py:1967:0: C0301: Line too long (119/100) (line-too-long)
views.py:1983:0: C0301: Line too long (111/100) (line-too-long)
views.py:1988:0: C0301: Line too long (112/100) (line-too-long)
views.py:2004:0: C0301: Line too long (111/100) (line-too-long)
views.py:2026:0: C0301: Line too long (108/100) (line-too-long)
views.py:2031:0: C0301: Line too long (111/100) (line-too-long)
views.py:2047:0: C0301: Line too long (108/100) (line-too-long)
views.py:2052:0: C0301: Line too long (111/100) (line-too-long)
views.py:2067:0: C0301: Line too long (108/100) (line-too-long)
views.py:2072:0: C0301: Line too long (111/100) (line-too-long)
views.py:2087:0: C0301: Line too long (108/100) (line-too-long)
views.py:2092:0: C0301: Line too long (111/100) (line-too-long)
views.py:1:0: C0302: Too many lines in module (2122/1000) (too-many-lines)
views.py:1:0: C0111: Missing module docstring (missing-docstring)
views.py:22:0: W0404: Reimport 'login' (imported line 3) (reimported)
views.py:26:0: W0401: Wildcard import BookingQBAApp.models (wildcard-import)
views.py:27:0: W0401: Wildcard import BookingQBAApp.serializers (wildcard-import)
views.py:32:0: W0401: Wildcard import BookingQBAApp.utils (wildcard-import)
views.py:37:0: C0111: Missing function docstring (missing-docstring)
views.py:54:0: C0111: Missing function docstring (missing-docstring)
views.py:56:8: W0622: Redefining built-in 'id' (redefined-builtin)
views.py:56:8: C0103: Variable name "id" doesn't conform to snake_case naming style (invalid-name)
views.py:62:12: C0103: Variable name "m" doesn't conform to snake_case naming style (invalid-name)
views.py:54:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
views.py:71:0: C0111: Missing function docstring (missing-docstring)
views.py:76:4: C0103: Variable name "rentModes" doesn't conform to snake_case naming style (invalid-name)
views.py:79:4: C0103: Variable name "referenceZones" doesn't conform to snake_case naming style (invalid-name)

views.py:95:0: C0111: Missing function docstring (missing-docstring)

views.py:115:0: C0111: Missing function docstring (missing-docstring)

views.py:116:4: C0103: Variable name "typesInfo" doesn't conform to snake_case naming style (invalid-name)

views.py:117:4: C0103: Variable name "typeInfo" doesn't conform to snake_case naming style (invalid-name)

views.py:119:8: C0103: Variable name "typeInfo" doesn't conform to snake_case naming style (invalid-name)

views.py:122:12: C0103: Variable name "lastChangeRateDate" doesn't conform to snake_case naming style (invalid-name)

views.py:123:12: C0103: Variable name "changeRate" doesn't conform to snake_case naming style (invalid-name)

views.py:130:0: C0111: Missing function docstring (missing-docstring)

views.py:157:0: C0111: Missing function docstring (missing-docstring)

views.py:161:0: C0111: Missing function docstring (missing-docstring)

views.py:167:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:177:0: C0111: Missing function docstring (missing-docstring)

views.py:183:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:184:12: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:205:0: C0111: Missing function docstring (missing-docstring)

views.py:210:0: C0111: Missing function docstring (missing-docstring)

views.py:210:0: R0914: Too many local variables (27/15) (too-many-locals)

views.py:211:4: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:212:4: C0103: Variable name "filterList" doesn't conform to snake_case naming style (invalid-name)

views.py:214:8: C0103: Variable name "rentMode" doesn't conform to snake_case naming style (invalid-name)

views.py:216:8: C0103: Variable name "referenceZone" doesn't conform to snake_case naming style (invalid-name)

views.py:238:8: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:241:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:244:12: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:245:11: R1714: Consider merging these comparisons with "in" to "municipality not in ('0', None)" (consider-using-in)

views.py:245:35: C0121: Comparison to None should be 'expr is not None' (singleton-comparison)

views.py:246:12: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:248:15: R1714: Consider merging these comparisons with "in" to "province not in ('0', None)" (consider-using-in)

views.py:248:35: C0121: Comparison to None should be 'expr is not None' (singleton-comparison)

views.py:249:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:250:15: R1714: Consider merging these comparisons with "in" to "referenceZone not in ('0', None)" (consider-using-in)

views.py:250:40: C0121: Comparison to None should be 'expr is not None' (singleton-comparison)

views.py:251:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:258:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:265:12: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:272:12: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:275:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:277:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:281:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:283:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:287:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:289:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:292:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:294:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:298:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:300:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:304:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:306:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:309:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:311:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:315:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:317:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:321:16: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:323:16: C0103: Variable name "msgError" doesn't conform to snake_case naming style (invalid-name)

views.py:326:4: C0103: Variable name "rentModes" doesn't conform to snake_case naming style (invalid-name)

views.py:329:4: C0103: Variable name "referenceZones" doesn't conform to snake_case naming style (invalid-name)

views.py:331:4: C0103: Variable name "rentListCount" doesn't conform to snake_case naming style (invalid-name)

views.py:210:0: R0912: Too many branches (46/12) (too-many-branches)

views.py:210:0: R0915: Too many statements (119/50) (too-many-statements)

views.py:343:0: C0111: Missing function docstring (missing-docstring)

views.py:344:4: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:345:4: C0103: Variable name "rentModes" doesn't conform to snake_case naming style (invalid-name)

views.py:348:4: C0103: Variable name "referenceZones" doesn't conform to snake_case naming style (invalid-name)

views.py:350:4: C0103: Variable name "rentListCount" doesn't conform to snake_case naming style (invalid-name)

views.py:363:0: C0111: Missing function docstring (missing-docstring)

views.py:364:4: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:365:4: C0103: Variable name "rentModes" doesn't conform to snake_case naming style (invalid-name)

views.py:368:4: C0103: Variable name "referenceZones" doesn't conform to snake_case naming style (invalid-name)

views.py:370:4: C0103: Variable name "rentListCount" doesn't conform to snake_case naming style (invalid-name)

views.py:384:0: C0111: Missing function docstring (missing-docstring)

views.py:385:4: C0103: Variable name "rentModes" doesn't conform to snake_case naming style (invalid-name)

views.py:387:4: C0103: Variable name "referenceZones" doesn't conform to snake_case naming style (invalid-name)

views.py:388:4: C0103: Variable name "rentList" doesn't conform to snake_case naming style (invalid-name)

views.py:394:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:394:0: C0111: Missing function docstring (missing-docstring)

views.py:397:4: C0103: Variable name "ip" doesn't conform to snake_case naming style (invalid-name)

views.py:405:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:405:0: C0111: Missing function docstring (missing-docstring)

views.py:408:4: C0103: Variable name "ip" doesn't conform to snake_case naming style (invalid-name)

views.py:408:4: W0612: Unused variable 'ip' (unused-variable)

views.py:416:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:416:0: C0111: Missing function docstring (missing-docstring)

views.py:416:0: R0914: Too many local variables (22/15) (too-many-locals)

views.py:417:4: C0103: Variable name "rentDetail" doesn't conform to snake_case naming style (invalid-name)

views.py:419:4: C0103: Variable name "ip" doesn't conform to snake_case naming style (invalid-name)

views.py:420:4: C0103: Variable name "havenHistory" doesn't conform to snake_case naming style (invalid-name)

views.py:422:8: C0103: Variable name "rentView" doesn't conform to snake_case naming style (invalid-name)

views.py:424:4: C0103: Variable name "rentModes" doesn't conform to snake_case naming style (invalid-name)

views.py:426:4: C0103: Variable name "referenceZones" doesn't conform to snake_case naming style (invalid-name)

views.py:430:4: C0103: Variable name "changeRate" doesn't conform to snake_case naming style (invalid-name)

views.py:433:8: C0103: Variable name "cr" doesn't conform to snake_case naming style (invalid-name)

views.py:441:4: C0103: Variable name "countComments" doesn't conform to snake_case naming style (invalid-name)

views.py:443:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:445:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:416:0: R0912: Too many branches (14/12) (too-many-branches)

views.py:480:0: C0111: Missing function docstring (missing-docstring)

views.py:519:0: C0111: Missing function docstring (missing-docstring)

views.py:546:0: C0111: Missing function docstring (missing-docstring)

views.py:548:8: C0103: Variable name "poiType" doesn't conform to snake_case naming style (invalid-name)

views.py:546:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:560:0: C0111: Missing function docstring (missing-docstring)

views.py:562:8: C0103: Variable name "rentId" doesn't conform to snake_case naming style (invalid-name)

views.py:575:12: C0103: Variable name "offerId" doesn't conform to snake_case naming style (invalid-name)

views.py:589:12: C0103: Variable name "offerId" doesn't conform to snake_case naming style (invalid-name)

views.py:560:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:597:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:597:0: C0111: Missing function docstring (missing-docstring)

views.py:597:0: R0914: Too many local variables (16/15) (too-many-locals)

views.py:599:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:600:8: C0103: Variable name "rentDetail" doesn't conform to snake_case naming style (invalid-name)

views.py:601:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:605:12: C0103: Variable name "changeRate" doesn't conform to snake_case naming style (invalid-name)

views.py:608:16: C0103: Variable name "cr" doesn't conform to snake_case naming style (invalid-name)

views.py:610:82: R0123: Comparison to literal (literal-comparison)

views.py:613:80: R0123: Comparison to literal (literal-comparison)

views.py:617:12: C0103: Variable name "countComments" doesn't conform to snake_case naming style (invalid-name)

views.py:619:12: C0103: Variable name "rentsRating" doesn't conform to snake_case naming style (invalid-name)

views.py:620:12: C0103: Variable name "rentsViews" doesn't conform to snake_case naming style (invalid-name)

views.py:621:12: C0103: Variable name "placeRating" doesn't conform to snake_case naming style (invalid-name)

views.py:622:12: C0103: Variable name "placeViews" doesn't conform to snake_case naming style (invalid-name)

views.py:646:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:646:0: C0111: Missing function docstring (missing-docstring)

views.py:647:4: C0103: Variable name "rentDetail" doesn't conform to snake_case naming style (invalid-name)

views.py:656:0: C0111: Missing function docstring (missing-docstring)

views.py:657:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:659:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:667:20: C0103: Variable name "im" doesn't conform to snake_case naming style (invalid-name)

views.py:668:20: C0103: Variable name "g" doesn't conform to snake_case naming style (invalid-name)

views.py:689:0: C0111: Missing function docstring (missing-docstring)

views.py:691:4: C0103: Variable name "ip" doesn't conform to snake_case naming style (invalid-name)

views.py:693:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:700:0: C0111: Missing function docstring (missing-docstring)

views.py:708:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:709:12: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:722:0: C0111: Missing function docstring (missing-docstring)

views.py:728:0: C0111: Missing function docstring (missing-docstring)

views.py:735:0: C0111: Missing function docstring (missing-docstring)

views.py:736:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:738:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:754:0: C0111: Missing function docstring (missing-docstring)

views.py:760:0: C0111: Missing function docstring (missing-docstring)

views.py:761:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:763:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:780:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)
views.py:780:0: C0111: Missing function docstring (missing-docstring)
views.py:782:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:784:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:799:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)
views.py:799:0: C0111: Missing function docstring (missing-docstring)
views.py:801:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:803:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:814:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:822:0: C0111: Missing function docstring (missing-docstring)
views.py:828:0: C0111: Missing function docstring (missing-docstring)
views.py:829:4: W0622: Redefining built-in 'list' (redefined-builtin)
views.py:835:8: W0702: No exception type(s) specified (bare-except)
views.py:847:0: C0111: Missing function docstring (missing-docstring)
views.py:848:4: W0622: Redefining built-in 'list' (redefined-builtin)
views.py:852:4: W0702: No exception type(s) specified (bare-except)
views.py:862:0: C0111: Missing function docstring (missing-docstring)
views.py:870:0: C0111: Missing function docstring (missing-docstring)
views.py:871:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:873:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:889:0: C0111: Missing function docstring (missing-docstring)
views.py:895:0: C0111: Missing function docstring (missing-docstring)
views.py:896:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:898:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:914:0: C0111: Missing function docstring (missing-docstring)
views.py:920:0: C0111: Missing function docstring (missing-docstring)
views.py:921:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:923:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:939:0: C0111: Missing function docstring (missing-docstring)
views.py:940:4: C0103: Variable name "drawTypes" doesn't conform to snake_case naming style (invalid-name)
views.py:945:0: C0111: Missing function docstring (missing-docstring)
views.py:946:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:948:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:964:0: C0111: Missing function docstring (missing-docstring)
views.py:965:4: C0103: Variable name "poiTypes" doesn't conform to snake_case naming style (invalid-name)
views.py:970:0: C0111: Missing function docstring (missing-docstring)
views.py:971:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:973:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:989:0: C0111: Missing function docstring (missing-docstring)

views.py:991:4: C0103: Variable name "poiType" doesn't conform to snake_case naming style (invalid-name)

views.py:996:0: C0111: Missing function docstring (missing-docstring)

views.py:997:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:999:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1015:0: C0111: Missing function docstring (missing-docstring)

views.py:1021:0: C0111: Missing function docstring (missing-docstring)

views.py:1022:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1024:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1031:12: E1120: No value for argument 'title' in function call (no-value-for-parameter)

views.py:1040:0: C0111: Missing function docstring (missing-docstring)

views.py:1046:0: C0111: Missing function docstring (missing-docstring)

views.py:1047:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1049:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1065:0: C0111: Missing function docstring (missing-docstring)

views.py:1074:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:1074:0: C0111: Missing function docstring (missing-docstring)

views.py:1090:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:1090:0: C0111: Missing function docstring (missing-docstring)

views.py:1101:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:1101:0: C0111: Missing function docstring (missing-docstring)

views.py:1107:0: C0111: Missing function docstring (missing-docstring)

views.py:1108:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1110:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1126:0: C0111: Missing function docstring (missing-docstring)

views.py:1129:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1126:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:1152:0: C0111: Missing function docstring (missing-docstring)

views.py:1158:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:1158:0: C0111: Missing function docstring (missing-docstring)

views.py:1182:0: C0111: Missing function docstring (missing-docstring)

views.py:1189:0: C0111: Missing function docstring (missing-docstring)

views.py:1196:0: C0111: Missing function docstring (missing-docstring)

views.py:1203:0: C0111: Missing function docstring (missing-docstring)

views.py:1206:4: C0103: Variable name "arrayReport" doesn't conform to snake_case naming style (invalid-name)

views.py:1207:8: C0103: Variable name "r" doesn't conform to snake_case naming style (invalid-name)

views.py:1208:11: C1801: Do not use `len(SEQUENCE)` to determine if a sequence is empty (len-as-condition)

views.py:1214:0: C0111: Missing function docstring (missing-docstring)

views.py:1221:0: C0111: Missing function docstring (missing-docstring)

views.py:1222:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1224:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1240:0: C0111: Missing function docstring (missing-docstring)

views.py:1246:0: C0111: Missing function docstring (missing-docstring)

views.py:1253:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:1253:0: C0111: Missing function docstring (missing-docstring)

views.py:1255:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1262:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)

views.py:1262:0: C0111: Missing function docstring (missing-docstring)

views.py:1282:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1289:0: C0111: Missing function docstring (missing-docstring)

views.py:1295:0: C0111: Missing function docstring (missing-docstring)

views.py:1298:4: C0103: Variable name "drawTypes" doesn't conform to snake_case naming style (invalid-name)

views.py:1299:4: C0103: Variable name "changeRate" doesn't conform to snake_case naming style (invalid-name)

views.py:1301:8: C0103: Variable name "dt" doesn't conform to snake_case naming style (invalid-name)

views.py:1303:12: C0103: Variable name "cr" doesn't conform to snake_case naming style (invalid-name)

views.py:1308:11: C0121: Comparison to False should be 'not expr' (singleton-comparison)

views.py:1310:4: R1705: Unnecessary "elif" after "return" (no-else-return)

views.py:1295:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:1317:0: C0111: Missing function docstring (missing-docstring)

views.py:1317:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:1323:0: C0111: Missing function docstring (missing-docstring)

views.py:1326:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1327:12: C0103: Variable name "drawTypes" doesn't conform to snake_case naming style (invalid-name)

views.py:1328:16: C0103: Variable name "dt" doesn't conform to snake_case naming style (invalid-name)

views.py:1329:16: C0103: Variable name "purchasePoint" doesn't conform to snake_case naming style (invalid-name)

views.py:1330:16: C0103: Variable name "salePoint" doesn't conform to snake_case naming style (invalid-name)

views.py:1323:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:1366:0: C0111: Missing function docstring (missing-docstring)

views.py:1372:0: C0111: Missing function docstring (missing-docstring)

views.py:1378:0: C0111: Missing function docstring (missing-docstring)

views.py:1379:4: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1381:8: R1705: Unnecessary "else" after "return" (no-else-return)

views.py:1402:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)
views.py:1402:0: C0111: Missing function docstring (missing-docstring)
views.py:1407:12: C0103: Variable name "s" doesn't conform to snake_case naming style (invalid-name)
views.py:1419:0: C0103: Argument name "pk" doesn't conform to snake_case naming style (invalid-name)
views.py:1419:0: C0111: Missing function docstring (missing-docstring)
views.py:1438:0: C0111: Missing function docstring (missing-docstring)
views.py:1440:8: W0622: Redefining built-in 'id' (redefined-builtin)
views.py:1440:8: C0103: Variable name "id" doesn't conform to snake_case naming style (invalid-name)
views.py:1438:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
views.py:1454:0: C0111: Missing function docstring (missing-docstring)
views.py:1456:8: W0622: Redefining built-in 'id' (redefined-builtin)
views.py:1456:8: C0103: Variable name "id" doesn't conform to snake_case naming style (invalid-name)
views.py:1466:0: C0111: Missing function docstring (missing-docstring)
views.py:1469:12: C0103: Variable name "n" doesn't conform to snake_case naming style (invalid-name)
views.py:1476:0: C0111: Missing function docstring (missing-docstring)
views.py:1481:0: C0111: Missing function docstring (missing-docstring)
views.py:1482:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:1484:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:1499:0: C0111: Missing function docstring (missing-docstring)
views.py:1504:0: C0111: Missing function docstring (missing-docstring)
views.py:1505:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:1507:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:1522:0: C0111: Missing function docstring (missing-docstring)
views.py:1527:0: C0111: Missing function docstring (missing-docstring)
views.py:1528:4: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:1530:8: R1705: Unnecessary "else" after "return" (no-else-return)
views.py:1546:0: C0111: Missing function docstring (missing-docstring)
views.py:1552:0: C0111: Missing function docstring (missing-docstring)
views.py:1569:0: C0111: Missing function docstring (missing-docstring)
views.py:1575:0: C0111: Missing function docstring (missing-docstring)
views.py:1592:0: C0111: Missing class docstring (missing-docstring)
views.py:1593:4: C0111: Missing method docstring (missing-docstring)
views.py:1600:25: E1101: Class 'Token' has no 'objects' member (no-member)
views.py:1609:16: C0103: Variable name "r" doesn't conform to snake_case naming style (invalid-name)
views.py:1600:15: W0612: Unused variable 'created' (unused-variable)
views.py:1593:4: R0201: Method could be a function (no-self-use)
views.py:1615:0: C0111: Missing class docstring (missing-docstring)
views.py:1616:4: C0111: Missing method docstring (missing-docstring)
views.py:1620:12: C0103: Variable name "userDeviceId" doesn't conform to snake_case naming style (invalid-name)

views.py:1623:16: C0103: Variable name "userDeviceNew" doesn't conform to snake_case naming style (invalid-name)

views.py:1625:15: C1801: Do not use `len(SEQUENCE)` to determine if a sequence is empty (len-as-condition)

views.py:1616:4: R0201: Method could be a function (no-self-use)

views.py:1632:0: C0111: Missing class docstring (missing-docstring)

views.py:1633:4: C0111: Missing method docstring (missing-docstring)

views.py:1637:12: C0103: Variable name "userDeviceId" doesn't conform to snake_case naming style (invalid-name)

views.py:1639:16: C0103: Variable name "userDeviceNew" doesn't conform to snake_case naming style (invalid-name)

views.py:1641:15: C1801: Do not use `len(SEQUENCE)` to determine if a sequence is empty (len-as-condition)

views.py:1643:20: C0103: Variable name "rentDetail" doesn't conform to snake_case naming style (invalid-name)

views.py:1633:4: R0201: Method could be a function (no-self-use)

views.py:1653:0: C0111: Missing class docstring (missing-docstring)

views.py:1654:4: C0111: Missing method docstring (missing-docstring)

views.py:1658:12: C0103: Variable name "userDeviceId" doesn't conform to snake_case naming style (invalid-name)

views.py:1660:16: C0103: Variable name "userDeviceNew" doesn't conform to snake_case naming style (invalid-name)

views.py:1662:15: C1801: Do not use `len(SEQUENCE)` to determine if a sequence is empty (len-as-condition)

views.py:1654:4: R0201: Method could be a function (no-self-use)

views.py:1680:0: C0111: Missing class docstring (missing-docstring)

views.py:1681:4: C0111: Missing method docstring (missing-docstring)

views.py:1685:12: C0103: Variable name "userDeviceId" doesn't conform to snake_case naming style (invalid-name)

views.py:1687:16: C0103: Variable name "userId" doesn't conform to snake_case naming style (invalid-name)

views.py:1689:20: C0103: Variable name "rentIdList" doesn't conform to snake_case naming style (invalid-name)

views.py:1691:28: C0103: Variable name "u" doesn't conform to snake_case naming style (invalid-name)

views.py:1693:36: C0103: Variable name "c" doesn't conform to snake_case naming style (invalid-name)

views.py:1683:8: R1702: Too many nested blocks (7/5) (too-many-nested-blocks)

views.py:1699:28: C0103: Variable name "u" doesn't conform to snake_case naming style (invalid-name)

views.py:1700:32: C0103: Variable name "c" doesn't conform to snake_case naming style (invalid-name)

views.py:1683:8: R1702: Too many nested blocks (6/5) (too-many-nested-blocks)

views.py:1705:24: C0103: Variable name "u" doesn't conform to snake_case naming style (invalid-name)

views.py:1706:28: C0103: Variable name "c" doesn't conform to snake_case naming style (invalid-name)

views.py:1711:16: C0103: Variable name "userDeviceNew" doesn't conform to snake_case naming style (invalid-name)

views.py:1713:20: C0103: Variable name "u" doesn't conform to snake_case naming style (invalid-name)

views.py:1714:24: C0103: Variable name "c" doesn't conform to snake_case naming style (invalid-name)

views.py:1693:36: W0612: Unused variable 'c' (unused-variable)

views.py:1681:4: R0201: Method could be a function (no-self-use)

views.py:1681:4: R0912: Too many branches (16/12) (too-many-branches)

views.py:1722:0: C0111: Missing function docstring (missing-docstring)

views.py:1722:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:1727:0: C0111: Missing function docstring (missing-docstring)

views.py:1727:28: W0613: Unused argument 'request' (unused-argument)

views.py:1732:0: C0111: Missing function docstring (missing-docstring)

views.py:1732:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)

views.py:1737:0: C0111: Missing function docstring (missing-docstring)

views.py:1737:36: W0621: Redefining name 'uuid' from outer scope (line 26) (redefined-outer-name)

views.py:1737:27: W0613: Unused argument 'request' (unused-argument)

views.py:1742:0: C0111: Missing function docstring (missing-docstring)

views.py:1742:19: W0613: Unused argument 'request' (unused-argument)

views.py:1757:0: C0111: Missing class docstring (missing-docstring)

views.py:1757:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1762:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1779:0: C0111: Missing class docstring (missing-docstring)

views.py:1779:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1785:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1801:0: C0111: Missing class docstring (missing-docstring)

views.py:1801:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1807:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1822:0: C0111: Missing class docstring (missing-docstring)

views.py:1822:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1828:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1832:12: C0103: Variable name "drawTypes_id" doesn't conform to snake_case naming style (invalid-name)

views.py:1833:12: C0103: Variable name "drawTypes_log" doesn't conform to snake_case naming style (invalid-name)

views.py:1843:0: C0111: Missing class docstring (missing-docstring)

views.py:1843:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1849:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1853:12: C0103: Variable name "poiTypes_id" doesn't conform to snake_case naming style (invalid-name)

views.py:1854:12: C0103: Variable name "poiTypes_log" doesn't conform to snake_case naming style (invalid-name)

views.py:1856:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:1864:0: C0111: Missing class docstring (missing-docstring)

views.py:1864:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1870:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1877:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:1885:0: C0111: Missing class docstring (missing-docstring)

views.py:1885:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1891:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1895:12: C0103: Variable name "changeRate_id" doesn't conform to snake_case naming style (invalid-name)

views.py:1896:12: C0103: Variable name "changeRate_log" doesn't conform to snake_case naming style (invalid-name)

views.py:1898:12: C0103: Variable name "lastChangeRate" doesn't conform to snake_case naming style (invalid-name)

views.py:1899:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:1908:0: C0111: Missing class docstring (missing-docstring)

views.py:1908:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1914:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1918:12: C0103: Variable name "rentMode_id" doesn't conform to snake_case naming style (invalid-name)

views.py:1919:12: C0103: Variable name "rentMode_log" doesn't conform to snake_case naming style (invalid-name)

views.py:1921:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:1929:0: C0111: Missing class docstring (missing-docstring)

views.py:1929:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1935:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1939:12: C0103: Variable name "referenceZone_id" doesn't conform to snake_case naming style (invalid-name)

views.py:1940:12: C0103: Variable name "referenceZone_log" doesn't conform to snake_case naming style (invalid-name)

views.py:1942:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:1951:0: C0111: Missing class docstring (missing-docstring)

views.py:1951:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1957:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1964:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:1972:0: C0111: Missing class docstring (missing-docstring)

views.py:1972:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1978:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:1985:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:1993:0: C0111: Missing class docstring (missing-docstring)

views.py:1993:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:1999:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:2006:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:2015:0: C0111: Missing class docstring (missing-docstring)

views.py:2015:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:2021:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:2028:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:2036:0: C0111: Missing class docstring (missing-docstring)

views.py:2036:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:2042:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:2049:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:2056:0: C0111: Missing class docstring (missing-docstring)

views.py:2056:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:2062:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:2069:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:2076:0: C0111: Missing class docstring (missing-docstring)

views.py:2076:0: R0901: Too many ancestors (12/7) (too-many-ancestors)

views.py:2082:11: C0121: Comparison to None should be 'expr is None' (singleton-comparison)

views.py:2089:16: C0103: Variable name "p" doesn't conform to snake_case naming style (invalid-name)

views.py:2109:0: C0111: Missing function docstring (missing-docstring)

views.py:10:0: W0611: Unused get_object_or_404 imported from django.shortcuts (unused-import)

views.py:12:0: W0611: Unused csrf imported from django.template.context_processors (unused-import)

views.py:14:0: W0611: Unused reverse_lazy imported from django.urls (unused-import)

views.py:22:0: W0611: Unused logout imported from django.contrib.auth as django_logout (unused-import)

views.py:23:0: W0611: Unused STATICFILES_DIRS imported from BookingQBA.settings (unused-import)

views.py:23:0: W0611: Unused BASE_DIR imported from BookingQBA.settings (unused-import)

views.py:26:0: W0614: Unused import os from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import uuid from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import datetime from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import DiaryViews from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import TotalViews from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import post_delete from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import post_save from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import pre_delete from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import receiver from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import collections from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import json from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import re from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import RequiredSecurityMixin from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import BaseDeleteView from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import menu_builder from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_letters from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_letters_may from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_letters_plus from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_letters_min from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_numbers from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_numbers_carne from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_letters_numbers from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_only_letters_numbers_plus from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_tcelular from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_tfijo from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_carne from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_nombre from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_apellidos from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_fecha from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_direccion from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_asunto from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_register_user from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_not_whitespace from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_passport_number from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import validate_cheque from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import get_list_json from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import fecha_value from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import resize_image from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import bd_create_500_record from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import folder_db_read from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import delete_address_db from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import analitic_rent_wish from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import emotion_average from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import BytesIO from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import timedelta from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import mean from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import Image from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import floatformat from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import messages from wildcard import (unused-wildcard-import)

views.py:26:0: W0614: Unused import DELETION from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import login_required from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import ContentType from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import SuccessMessageMixin from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import ValidationError from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import HttpResponse from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import RequestContext from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import get_template from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import reverse from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import force_str from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import View from wildcard import (unused-wildcard-import)
views.py:26:0: W0614: Unused import settings from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import base64 from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import safe_string from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import exceptions from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import ContentFile from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import File from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import strip_tags from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import html_safe from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import basestring from wildcard import (unused-wildcard-import)
views.py:27:0: W0614: Unused import serializers from wildcard import (unused-wildcard-import)
views.py:28:0: C0411: third party import "from notifications.signals import notify" should be placed before "from BookingQBA.settings import STATICFILES_DIRS, BASE_DIR" (wrong-import-order)
views.py:29:0: C0411: third party import "from notifications import models as models_notify" should be placed before "from BookingQBA.settings import STATICFILES_DIRS, BASE_DIR" (wrong-import-order)
views.py:33:0: C0411: third party import "import sweetify" should be placed before "from BookingQBA.settings import STATICFILES_DIRS, BASE_DIR" (wrong-import-order)
views.py:20:0: C0412: Imports from package django are not grouped (ungrouped-imports)
views.py:31:0: C0412: Imports from package BookingQBAapp are not grouped (ungrouped-imports)

Your code has been rated at 5.67/10 (previous run: 5.67/10, +0.00)