PROYECTO
BUSCOCHOZA.COM

DOCUMENTACIÓN EXTERNA

INTEGRANTES:

ESTEBAN AGUILAR VALVERDE
BAYRON PORTUGUEZ CASTILLO
SILVIA SEGURA SOTO

1. Resumen Ejecutivo

Buscar casa, apartamento y demás para alquilar o comprar es cansado y un poco fastidioso ya que no se cuenta con una plataforma que muestre de forma ordenada y clara la información deseada para realizar la compra, por lo que se propone la solución realizando una aplicación la cual será un buscador de casas de alquiler y casas a la venta, en el cual los usuarios autenticados podrán buscar y realizar publicaciones con la información necesaria ya sea para casas de alquiler o venta en distintos sitios del territorio costarricense.

Los usuarios podrán ubicar casas de alquiler y de compra por medio de un enlace Web que podrá ser accedido desde distintos dispositivos tanto de escritorio como móviles. Los datos serán almacenados en una base de datos que se encontrará en la nube.
Se realizarán conexiones a API's externos con los que la aplicación podrá auto complementarse y así brindar más eficiencia y usabilidad a los usuarios.

Los API's serán Google Maps y Facebook.

2. Especificación de Requerimientos

2.1. Introducción

2.1.1. Propósito

Identificar, analizar y definir las características que debe satisfacer el Sistema para buscar casas de alquiler o compra de modo que los usuarios y personal de tecnologías tengan visibilidad e iguales expectativas del producto que se desarrollará.

2.1.2. Alcance

Este documento define las características funcionales y no funciona-les de alto nivel que se derivan de reuniones del equipo de trabajo para definir el objetivo de la aplicación.

2.1.3. Visión General

Este documento es la visión de la solución tecnológica que facilitará el proceso de búsqueda de casas de alquiler y venta y también la venta o alquiler de casas.

2.2. Posicionamiento

2.2.1. Declaración del Problema

Para todo usuario que busque casa para alquilar o comprar siempre ha sido un problema, esto debido que es difícil poder saber cuáles casas están disponibles para alquilar o vender en un lugar específico.

2.2.2. Declaración del posicionamiento del producto

El nuevo sistema permitirá los usuarios generar búsquedas de casas en un lugar específico por medio de google maps, ingresar nuevas casas para poder vender o alquilar. Además de poderse loguear por medio de su cuenta de Facebook.

2.3. Descripciones de afectados y usuarios

Esta sección provee un perfil de los afectados y usuarios involucrados relacionados con el sistema, y los problemas clave que ellos perciben que deben ser abordados por la solución propuesta

2.3.1. Resumen de interesados

Usuarios	Responsabilidades
Administrador	- Administrar los usuarios - Administrar las publicaciones de las casas - Administrar búsquedas
Usuario-Básico	- Agregar publicaciones de casas para vender - Agregar publicaciones de casas para alquilar - Buscar casas en un lugar específico para alquilar o comprar.

2.3.2. Ambiente del usuario

El usuario podrá acceder a la aplicación por medio de la red. Lo podrá realizar mediante su computadora o cualquier dispositivo móvil con acceso a internet.

2.4. Visión general del producto

2.4.1. Perspectiva del producto

La aplicación se conectarán con dos API's externos, los cuales serán Google Maps y Facebook.

2.4.2. Necesidades y características del producto

El sistema deberá poseer una eficiente conexión con google maps y Facebook.

2.5. Característica del producto

2.5.1. Gestión de cuentas de usuario

El sistema deberá proveer las acciones básicas de agregar, modificar, eliminar una cuenta de usuario
La información general:

- Nombre
- Contraseña
- Correo
- Teléfono
- Dirección

Los roles encargado de realizar estas acciones son el administrador y el usuario.

2.5.1.1. Ingreso de nuevas cuentas de usuario

El usuario podrá acceder a la aplicación por medio de su cuenta de Facebook o podrá ingresar su nueva cuenta desde la aplicación.

2.5.1.2. Eliminación de cuentas de usuario

El administrador podrá eliminar cualquier cuenta de usuario.
El usuario podrá eliminar su propia cuenta de usuario desde la aplicación.

2.5.2. Gestión de Publicaciones

El sistema deberá proveer las acciones básicas de agregar y eliminar una publicación.
Dicha publicación se compone de una instalación ya sea casa, oficina o apartamento, al igual que el tipo de publicación (vender o alquilar)
La información general:

- Descripción
- Metros en Construcción
- Metros de Terreno
- Precio
- Fecha de Publicación
- Estado

Los roles encargado de realizar estas acciones son el administrador y el usuario.

2.5.2.1. Ingreso de una nueva publicación

El usuario podrá ingresar una nueva casa desde la aplicación y deberá especificar el tipo de instalación que es (casa, habitación, edificio, piso, oficina) y si es de compra o venta.

2.5.2.2. Eliminar publicación

El usuario podrá eliminar sus propias casas.

2.5.3. Búsqueda de casas

El sistema deberá de proveer la acción de búsqueda de casas.
El usuario podrá realizar búsquedas específicas de alquiler o compra en un lugar específico.

3. Prioridades

3.1. Funcionalidades

El sistema debe contar con las siguientes funciones:

3.1.1. Cuentas de usuario

El usuario puede loguearse de dos maneras:

3.1.1.1. Login por medio de Facebook

El sistema brindará la opción para que el usuario pueda ingresar a la aplicación por medio de sus credenciales de Facebook, esto con el fin de simplificar la utilización de la aplicación al no ingresar una nueva cuenta.

3.1.1.2. Cuenta de usuario

El usuario podrá ingresar a la aplicación mediante su propia cuenta ingresada directamente a la aplicación.

3.1.2. Búsqueda de casas

La aplicación brinda la opción de buscar casas de alquiler o venta por medio de google maps, en un lugar específico.

3.1.3. Publicación de casas

El usuario puede ingresar una nueva publicación la cual será de una instalación (edificio, casa, piso, apartamento, oficina) ya sea de alquiler o de venta.

3.2. Factores de calidad más importantes

La aplicación BuscoChoza deberá cumplir con los siguientes factores de calidad:

3.2.1. Usabilidad

La usabilidad es el factor más importante de la aplicación esto debido a que su objetivo es obtener una buena interacción con el usuario para poder facilitarle la búsqueda de casas.
La aplicación posee un buen diseño de HCI, con una interfaz bastante llamativa, intuitiva y funcional.

3.2.2. Eficiente

El segundo factor de la aplicación es la eficiencia debido a que debe poseer un tiempo de respuesta mínimo, por lo que debe realizar un uso razonable de recursos.

3.2.3. Mantenibilidad

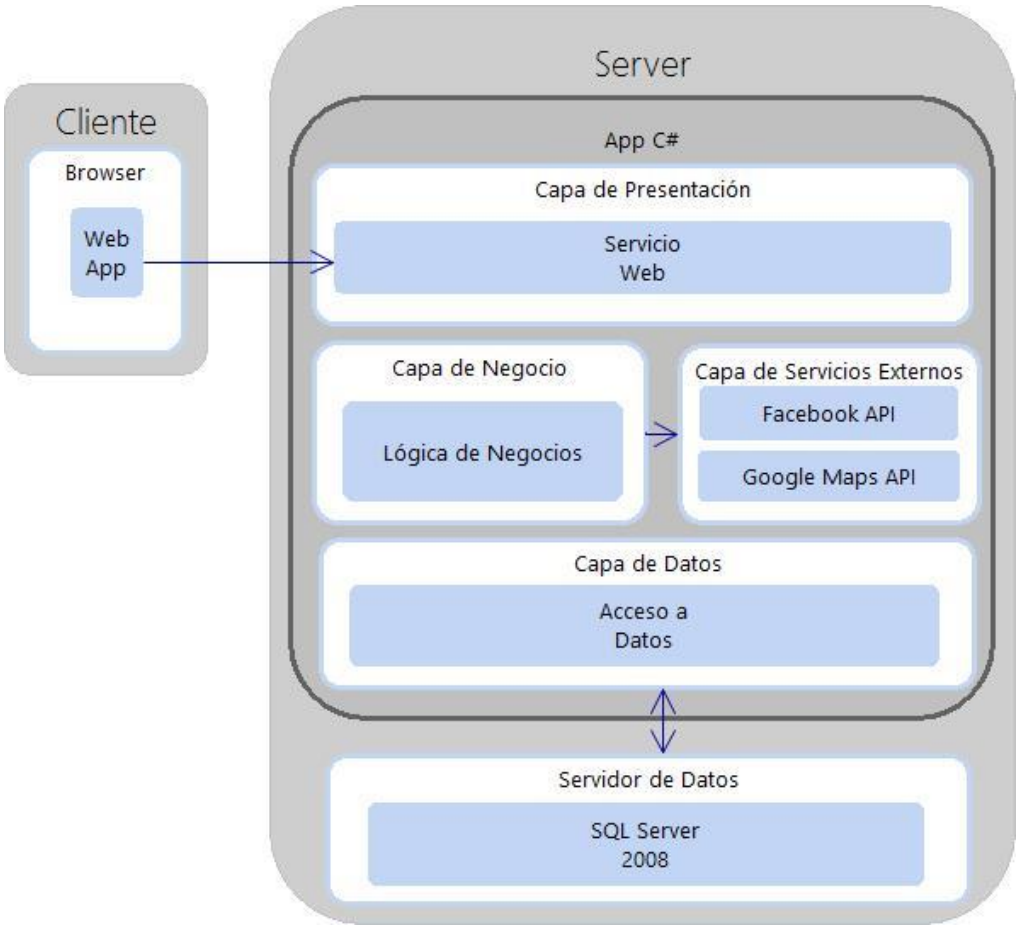
El factor de calidad mantenibilidad dentro de la aplicación no es tan importante, esto debido a que es muy específica con su objetivo por lo que no debe estar actualizándose ya que los usuarios estarán satisfechos con las funciones que esta realiza por lo que sufre de muy pocos cambios.

3.2.4. Confiabilidad

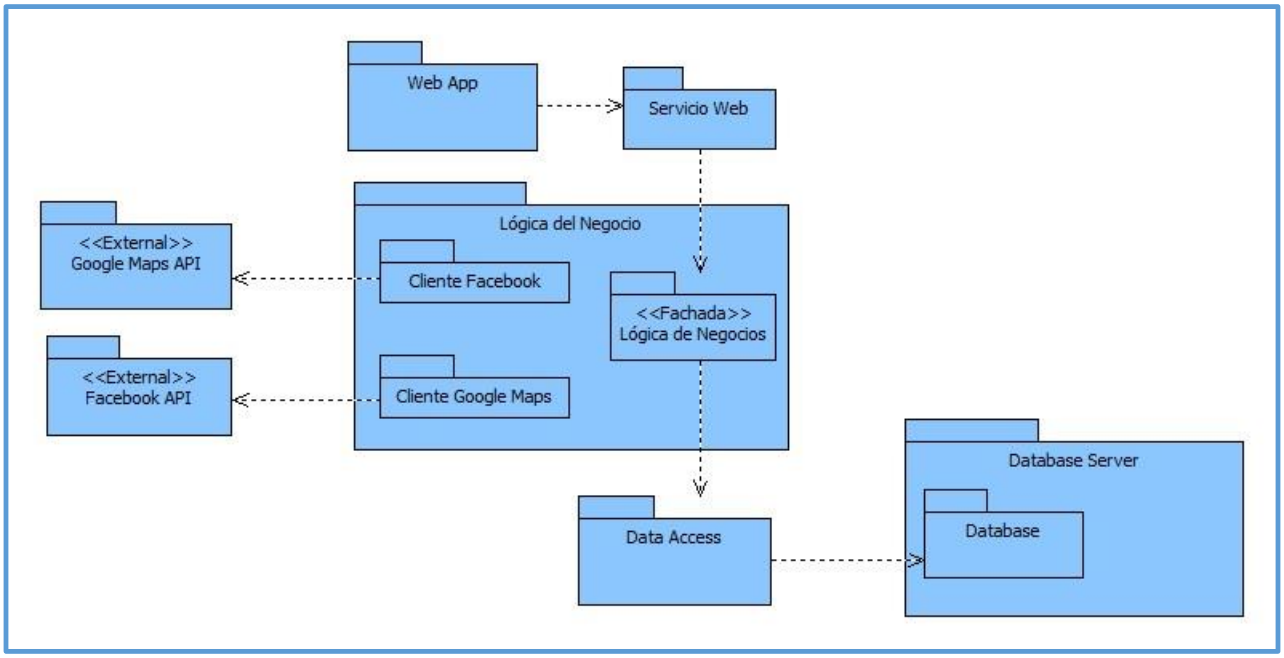
Por último tenemos la confiabilidad ya que al ser una aplicación web solamente informativa el usuario no realiza ninguna compra o alquiler por medio de la aplicación, la seguridad no es algo importante para la aplicación.

4. Descripción de diseño de alto nivel

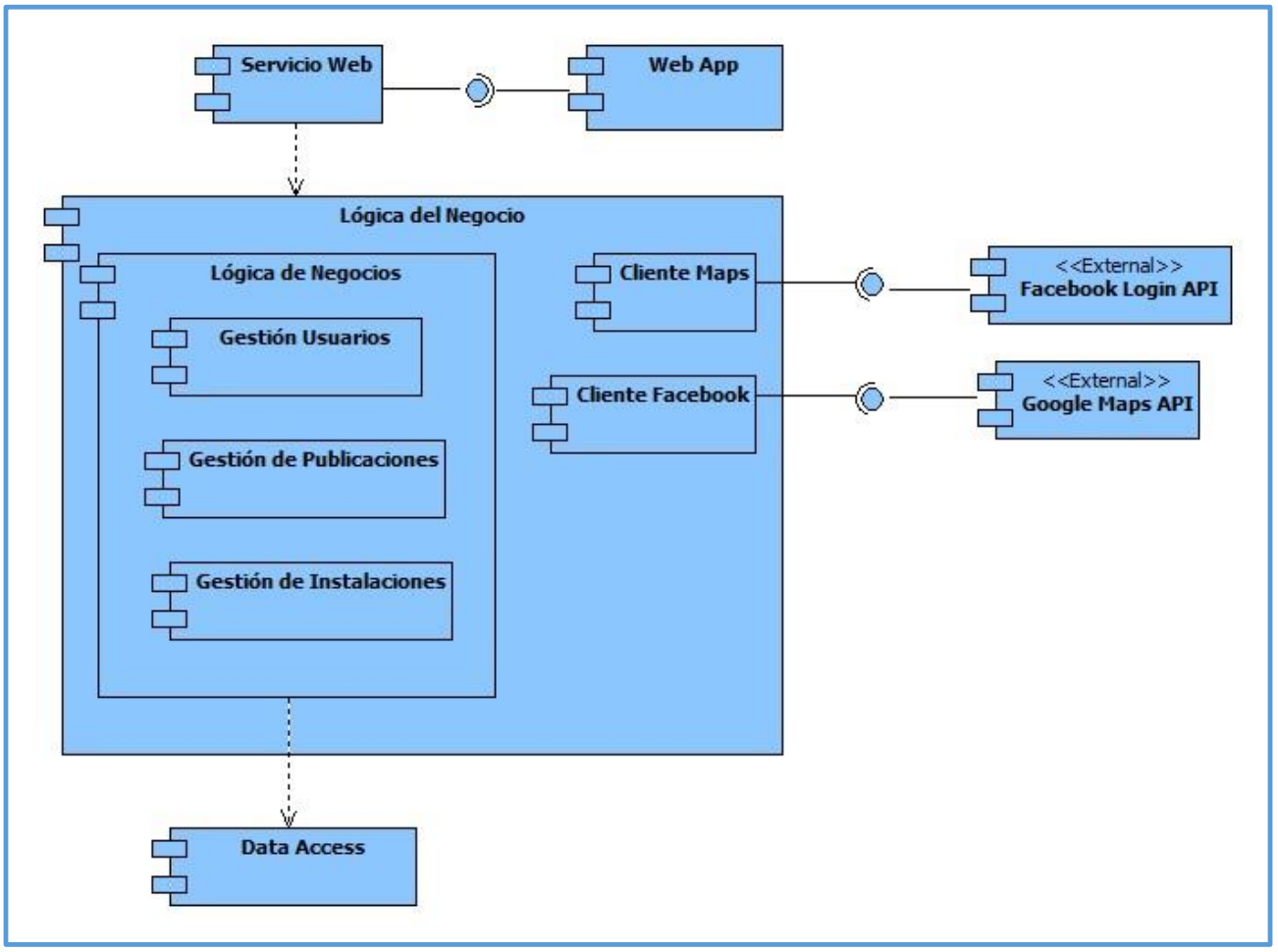
4.1. Diagrama de arquitectura conceptual



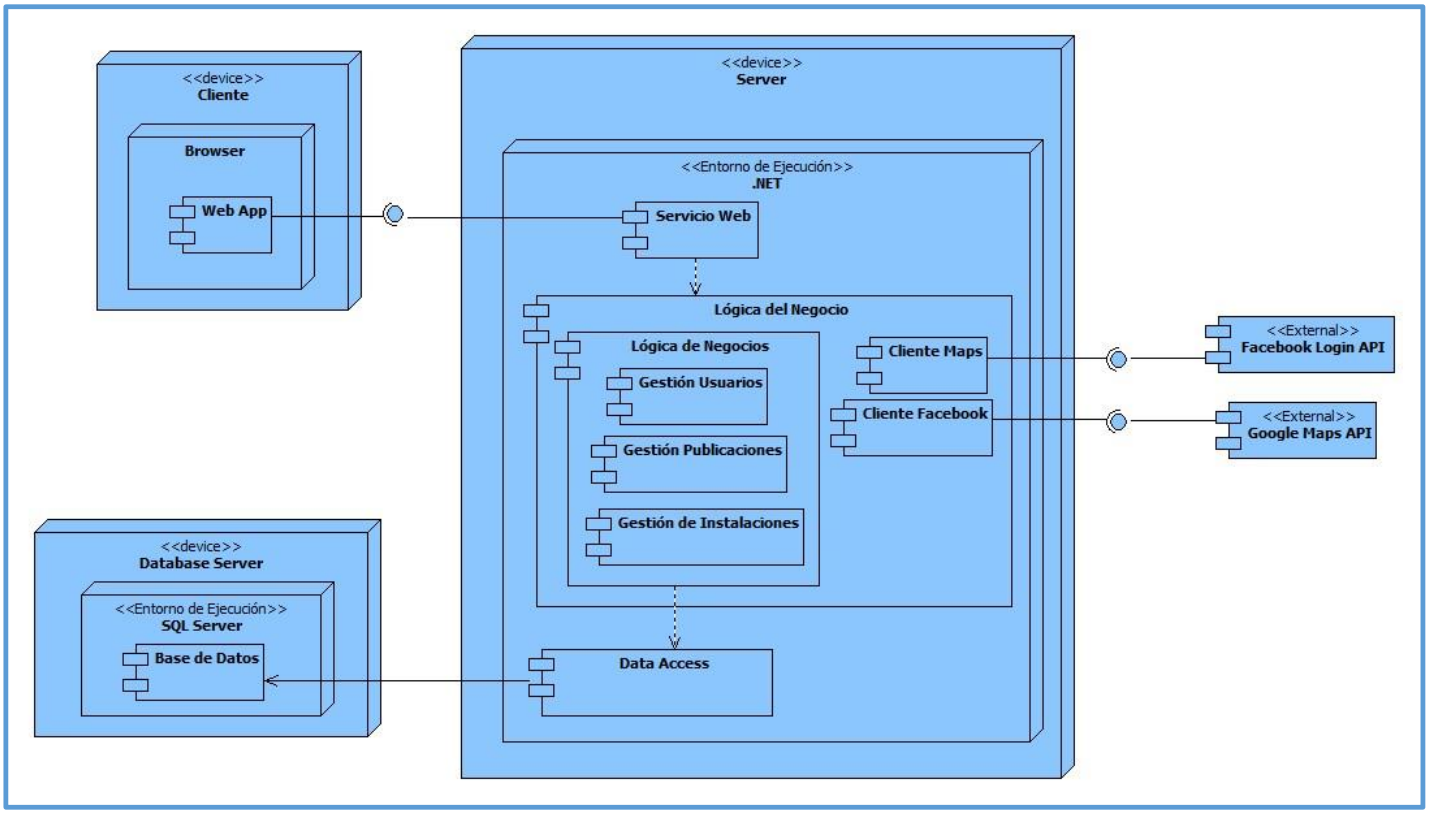
4.2. Diagrama de Paquetes



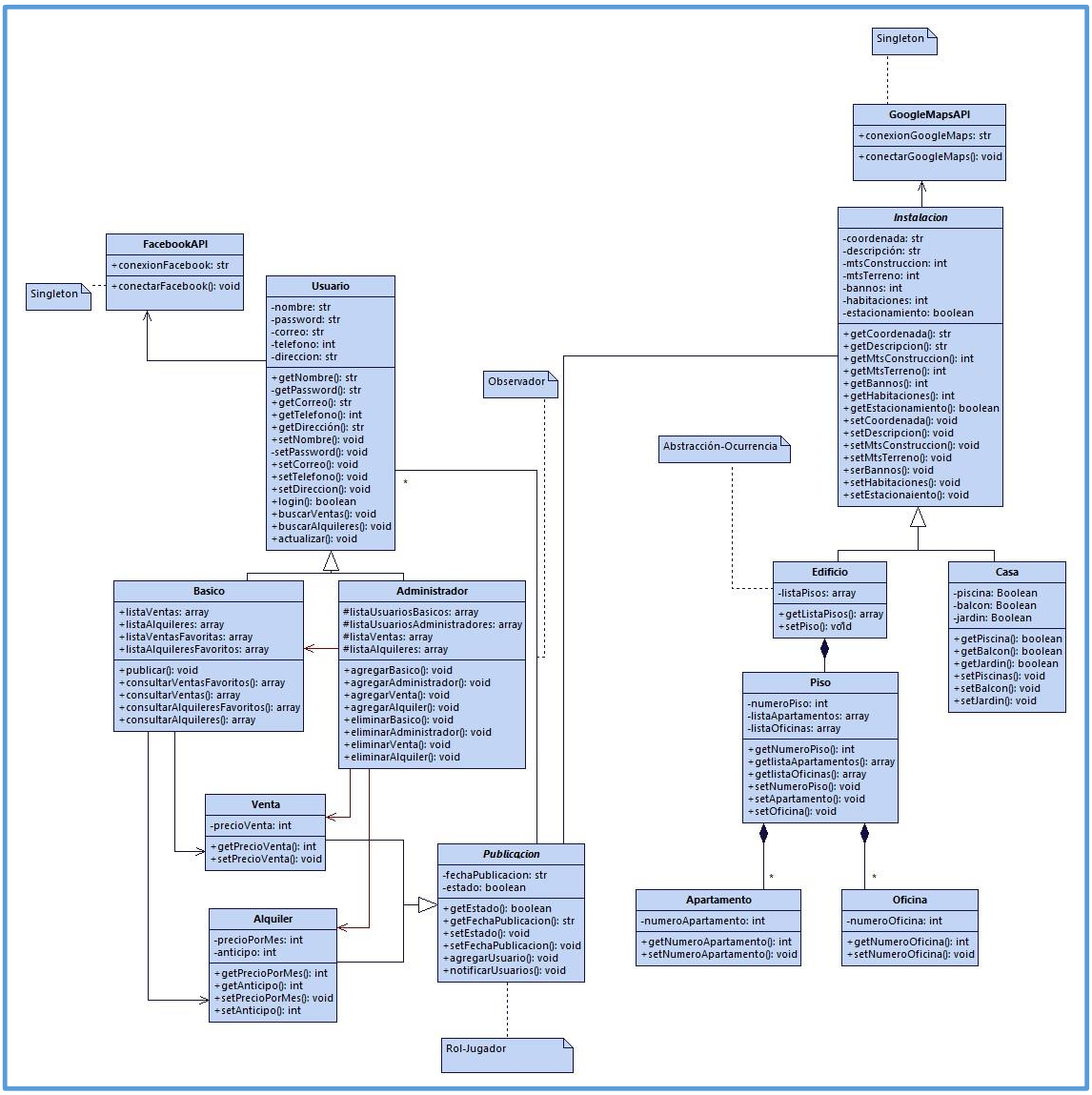
4.3. Diagrama de Componentes



4.4. Diagrama de Despliegue



4.5. Diagrama de Clases



5. Descripción detallada

A continuación se detallarán las clases diseñadas para la solución del problema planteado:

5.1. Usuario

Esta clase fue diseñada para el manejo general de usuarios en el sistema y para definir el estándar de información que requiere el sistema. Esta posee las características básicas que contendrán todos los objetos usuario independientemente del rol o función que cumpla. Se implementará el método de login que será igual para todos los usuarios.

Nombre	Usuario
Tipo	Concreta

Atributos

Visibilidad	Nombre	Tipo de dato
Public	nombre	String
Private	password	String
Public	correo	String
Public	telefono	Integer
Public	direccion	String

Métodos

Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getNombre	-	-	String
Private	getPassword	-	-	String
Public	getCorreo	-	-	String
Public	getTelefono	-	-	Integer
Public	getDireccion	-	-	String
Public	setNombre	nombre	String	Void
Private	setPassword	password	String	Void
Public	setCorreo	correo	String	Void
Public	setTelefono	telefono	Integer	Void
Public	setDireccion	direccion	String	Void
Public	login	nombre password	String String	Boolean
Public	buscarVentas	direccion	String	Void
Public	buscarAlquileres	direccion	String	Void
Public	Actualizar	-	-	Void

Dependencias	Administrador Basico
Padres	-
Implementación	FacebookAPI

5.2. Administrador

Esta clase tiene por objetivo el control y manejo general de sistema. Este usuario será definido a nivel interno en primera instancia. Esta clase hereda los atributos de usuario y además puede hacer lo mismo que un usuario básico, razón por la cual se aplicó el patrón jerarquía general (explicado a detalle en el documento Justificación de Patrones Utilizados).

Nombre	Administrador
Tipo	Concreta

Atributos		
Visibilidad	Nombre	Tipo de dato
Protected	listaUsuariosBasicos	array
Protected	listaUsuariosAdministradores	array
Protected	listaVentas	array
Protected	listaAlquileres	Array

Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	agregarBasico	-	-	Void
Public	agregarAdministradr	-	-	Void
Public	agregarVenta	-	-	Void
Public	eliminarBasico	-	-	Void
Public	eliminarAdministrador	-	-	Void
Public	eliminarVenta	-	-	Void
Public	eliminarAlquiler	-	-	Void

Dependencias	-
Padres	Usuario
Implementación	-

5.3. Basico

Esta clase tiene por objetivo el manejo de los usuarios regulares del sistema. Esta clase hereda de la clase Usuario para poder utilizar los atributos y métodos propios de esta clase. Además de hacer las búsquedas esenciales para el sistema.

Nombre	Basico
Tipo	Concreta

Atributos		
Visibilidad	Nombre	Tipo de dato
Public	nombre	String
Private	password	String
Public	correo	String
Public	telefono	Integer
Public	direccion	String
Public	listaVentas	Array
Public	listaAlquileres	Array
Public	listaVentasFavoritas	Array
Public	listaAlquileresFavoritos	Array

Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getNombre	-	-	String
Private	getPassword	-	-	String
Public	getCorreo	-	-	String
Public	getTelefono	-	-	Integer
Public	getDireccion	-	-	String
Public	setNombre	nombre	String	Void
Private	setPassword	password	String	Void
Public	setCorreo	correo	String	Void
Public	setTelefono	telefono	Integer	Void
Public	setDireccion	direccion	String	Void
Public	login	nombre password	String String	Boolean
Public	buscarVentas	direccion	String	Void
Public	buscarAlquileres	direccion	String	Void
Public	publicar	tipo coordenada informacion	String String String	Void
Public	consultarVentasFavorito	-	-	Void

Public	consultarVentas	-	-	Void
Public	consultarAlqui-leresFavorito	-	-	Void
Public	consultarAlqui-leres	-	-	Void
Dependencias			-	
Padres			Usuario	
Implementación			-	

5.4. Instalación

Esta clase tiene por objetivo regular la información que el sistema brindará acerca de los diversos tipos de instalación que el sistema pueda ofrecer. Esta clase posee los atributos y métodos básicos que tanto Edificio como Casa poseen. Esto brindara al sistema una mayor estandarización de la información que se maneja.

Nombre	Instalación
Tipo	Abstracta

Atributos		
Visibilidad	Nombre	Tipo de dato
Public	latitud	Float
Public	longitud	Float
Public	descripcion	String
Public	mtsConstruccion	String
Public	mtsTerreno	Integer
Public	bannos	Integer
Public	habitaciones	Integer
Public	estacionamiento	Boolean
Public	numeroDistrito	Int
Public	numeroProvincia	Int
Public	numeroCanton	int

Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getLatitud	-	-	Float
Public	getLongitud	-	-	Float
Public	getDescripcion	-	-	String
Public	getMtsConstruccion	-	-	String
Public	getMtsTerreno	-	-	Integer
Public	getBannos	-	-	Integer
Public	getHabitaciones	-	-	Integer
Public	getEstacionamiento	-	-	Boolean
Public	getNumeroDistrito	-	-	int
Public	getNumeroProvincia	-	-	int
Public	getNumeroCanton	-	-	Int
Public	setLatitud	latitud	Integer	Void
Public	setDescripcion	descripción	String	Void
Public	setMtsConstruccion	mtsConstruccion	Integer	Void
Public	setMtsTerreno	mtsTerreno	Integer	Void
Public	setBannos	bannos	Integer	Void
Public	setHabitaciones	habitaciones	Integer	Void

Public	setEstacionamiento	estacionamiento	Integer	Void
Public	setNumeroProvincia	numeroProvin-cia	Integer	Void
Public	setNumeroDistrito	numeroDistrito	Integer	Void
Public	setNumeroCanton	numeroCanton	Integer	Void
Dependencias		Casa Edificio		
Padres		-		
Implementación		GoogleMapsAPI		

5.5. Casa

Esta clase es un tipo particular de Instalación orientado más a viviendas. Esta clase hereda de la clase Instalación para contar con los atributos y métodos definidos en esta y además almacena información particular que a un comprador de una casa podría necesitar saber.

Nombre	Casa
Tipo	Concreta

Atributos		
Visibilidad	Nombre	Tipo de dato
Public	descripcion	String
Public	mtsConstruccion	String
Public	mtsTerreno	Integer
Public	bannos	Integer
Public	habitaciones	Integer
Public	estacionamiento	Boolean
Public	piscina	Boolean
Public	balcon	Boolean
Public	jardin	Boolean

Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getDescripcion	-	-	String
Public	getMtsConstruccion	-	-	String
Public	getMtsTerreno	-	-	Integer
Public	getBannos	-	-	Integer
Public	getHabitaciones	-	-	Integer
Public	getEstacionamiento	-	-	Boolean
Public	setDescripcion	descripción	String	Void
Public	setMtsConstruccion	mtsConstruccion	Integer	Void
Public	setMtsTerreno	mtsTerreno	Integer	Void
Public	setBannos	bannos	Integer	Void
Public	setHabitaciones	habitaciones	Integer	Void
Public	setEstacionamiento	estacionamiento	Integer	Void

Dependencias	
Padres	
Implementación	

5.6. Edificio

Esta clase es un tipo particular de Instalación orientado más a edificios apartamentales u oficentros. Esta clase hereda de la clase Instalación para contar con los atributos y métodos definidos en esta y además almacena información particular que a un comprador de un edificio podría necesitar saber. Esta clase cuenta con una lista de pisos para que el usuario se enteré de las dimensiones de la edificación y a su vez de otros detalles que se explicaran en las siguientes clases.

Nombre	Edificio
Tipo	Concreta

Atributos		
Visibilidad	Nombre	Tipo de dato
Public	descripcion	String
Public	mtsConstruccion	String
Public	mtsTerreno	Integer
Public	bannos	Integer
Public	habitaciones	Integer
Public	estacionamiento	Boolean
Public	listaPisos	Array[Pisos]

Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getDescripcion	-	-	String
Public	getMtsConstruccion	-	-	String
Public	getMtsTerreno	-	-	Integer
Public	getBannos	-	-	Integer
Public	getHabitaciones	-	-	Integer
Public	getEstacionamiento	-	-	Boolean
Public	setDescripcion	descripción	String	Void
Public	setMtsConstruccion	mtsConstruccion	Integer	Void
Public	setMtsTerreno	mtsTerreno	Integer	Void
Public	setBannos	bannos	Integer	Void
Public	setHabitaciones	habitaciones	Integer	Void
Public	setEstacionamiento	estacionamiento	Integer	Void
Public	getlistaPisos	-	-	Array[Piso]
Public	setPiso	piso	Piso	Void

Dependencias	Piso
Padres	Instalación
Implementación	-

5.7. Oficina

Esta clase es un tipo particular de habitación dentro de un piso de un edificio. Esta clase maneja atributos y métodos particulares que un usuario podría estar interesado en saber acerca de un edificio por la particularidad de una oficina y su aplicación.

Nombre				Oficina
Tipo				Concreta
Atributos				
Visibilidad		Nombre		Tipo de dato
Public		numeroOficina		Integer
Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getNumeroOficina	-	-	Integer
Public	setNuemeroOfi-cina	numeroOficina	Integer	Void
Dependencias				-
Padres				-
Implementación				-

5.8. Piso

Esta clase está asociada con la clase Edificio para manejar lo que posee cada nivel del edificio. Esta ayudara a detallar de forma más concreta donde está situado ya se un Apartamento o una Oficina y cuantos posee. Esto ayudara a manejar la información y evitar la replicación de información, esto pues su implantación esta aplicada con el patrón abstracción-ocurrencia (esto se detallara en el documento Justificación de patrones utilizados)

Nombre	Piso
Tipo	Concreta

Atributos		
Visibilidad	Nombre	Tipo de dato
Pubico	numeroPiso	Integer
Public	listaApartamento	Array[Apartamento]
Public	listaOficinas	Array[Oficina]

Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getNumeroPiso	-	-	String
Private	getlistaApartamento	-	-	String
Public	getListaOficinas	-	-	String
Public	setNumeroPiso	numeroPiso	Integer	Void
Public	setApartamento	numeroApartamento	Integer	Void

Dependencias	Apartamento Oficina
Padres	Edificio
Implementación	-

5.9. Apartamento

Esta clase es un tipo particular de habitación dentro de un piso de un edificio. Esta clase maneja atributos y métodos particulares que un usuario podría estar interesado en saber acerca de un edificio por la particularidad de un apartamento y su aplicación.

Nombre				Apartamento
Tipo				Concreta
Atributos				
Visibilidad		Nombre		Tipo de dato
Pubico		_numeroApartamento		Int
Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getNume-roApartamento	-	-	Integer
Public	setNume-roApartamento	numeroAparta-mento	Integer	Void
Dependencias				-
Padres				-
Implementación				-

5.10. Publicacion

Esta clase maneja las publicaciones específicas de un usuario del sitema.

Nombre			Publicacion	
Tipo			Abstracta	
Atributos				
Visibilidad	Nombre		Tipo de dato	
Public	fechaPublicacion		String	
Public	estado		Boolean	
Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getEstado	-	-	Boolean
Public	getFechaPublicacion	-	-	String
Public	setEstado	estado	Boolean	Void
Public	setFechaPublicacion	fecha	String	Void
Public	agregarUsuario	Usuario	usuario	Void
Public	notificarUsuario	-	-	Void
Dependencias			Venta Alquiler	
Padres			-	
Implementación			-	

5.11. Venta

Esta clase maneja la venta de instalaciones estas crearán las instalaciones posterior a la ubicación de la misma.

Nombre	Venta
Tipo	Concreta

Atributos		
Visibilidad	Nombre	Tipo de dato
Public	fechaPublicacion	String
Public	estado	Boolean
Public	precioVenta	Integer

Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getEstado	-	-	Boolean
Public	getFechaPublicacion	-	-	String
Public	setEstado	estado	Boolean	Void
Public	setFechaPublicacion	fecha	String	Void
Public	getPrecioVenta	-	-	Integer
Public	setPrecioVenta	precioVenta	Integer	Void

Dependencias	-
Padres	Publicación
Implementación	-

5.12. Alquiler

Esta clase maneja el alquiler de instalaciones estas crearán las instalaciones posterior a la ubicación de la misma.

Nombre			Alquiler	
Tipo			Concreta	
Atributos				
Visibilidad	Nombre		Tipo de dato	
Public	fechaPublicacion		String	
Public	estado		Boolean	
Public	precioPorMes		Integer	
Public	anticipo		Integer	
Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	getEstado	-	-	Boolean
Public	getFechaPublicacion	-	-	String
Public	setEstado	estado	Boolean	Void
Public	setFechaPublicacion	fecha	String	Void
Public	getAnticipo	-	-	Integer
Public	getPrecioPorMes	-	-	Integer
Public	setAnticipo	anticipo	Integer	Void
Public	setPrecioPorMes	precioPorMes	Integer	Void
Dependencias			-	
Padres			Publicacion	
Implementación			-	

5.13. FacebookAPI

Esta clase maneja la conexión a Facebook.

Nombre			FacebookAPI	
Tipo			Interface	
Atributos				
Visibilidad		Nombre		Tipo de dato
Public		conexionFacebook		String
Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	conectarFacebook	stringConexion	String	Void
Dependencias			-	
Padres			-	
Implementación			-	

5.14. GoogleMapsAPI

Esta clase maneja la conexión a Google Maps.

Nombre			FacebookAPI	
Tipo			Interface	
Atributos				
Visibilidad		Nombre		Tipo de dato
Public		conexionGoogleMaps		String
Métodos				
Visibilidad	Nombre	Parámetros	Tipo de dato	Tipo de retorno
Public	conectarGoogleMaps	stringConexion	String	Void
Dependencias			-	
Padres			-	
Implementación			-	

6. Justificación de Patrones

6.1. Singleton

El patrón se representa en las APIs tanto de Facebook como el de Google maps, al ingresar a la página web buscochoza.info se da la opción de ingresar desde la cuenta de Facebook si es que el usuario posee una, esto creará una instancia de esta clase y será la única existente, ya que solo se puede iniciar desde una cuenta de Facebook.

Sucede lo mismo con el API de google maps donde una vez iniciada la sesión, ya sea creando una cuenta nueva o mediante el API de Facebook, buscochoza.info desplegará un mapa el cual tendrá diferentes funcionalidades para el uso de la página web. Esta clase tendrá una sola instancia ya que solo se crea una conexión para mostrar un mapa en pantalla.

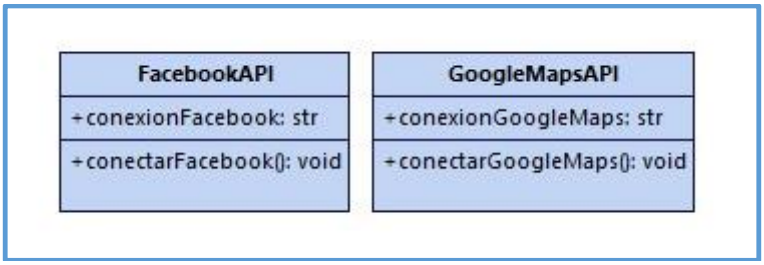


Figura 1 Patrón Singleton

6.2. Jugador Rol

Una instalación la cual puede ser de diferentes tipos (casa, apartamento, oficina) le pedirá al usuario que esté realizando una publicación, la forma en que se le dará publicidad a la instalación, existen 2 tipos de roles:

Venta: Este rol contará con un precio total por la venta de la casa, las instalaciones que tengan este rol se agruparan para que el usuario que este en búsqueda, si selecciona en el menú la opción "venta", se desplegaran en el mapa marcadores en las diferentes ubicaciones donde existan instalaciones que estén en venta.

Alquiler: Este rol cuenta con un precio mensual por el alquiler de la instalación así como un anticipo, las instalaciones que cuenten con este rol se agruparan para que un usuario que este en búsqueda, si selecciona la opción "alquiler".

La idea de cada rol es poder diferenciar a las instalaciones para que el momento de que el usuario realice búsquedas pueda encontrar la instalación con las características deseadas de manera más sencilla y veloz.

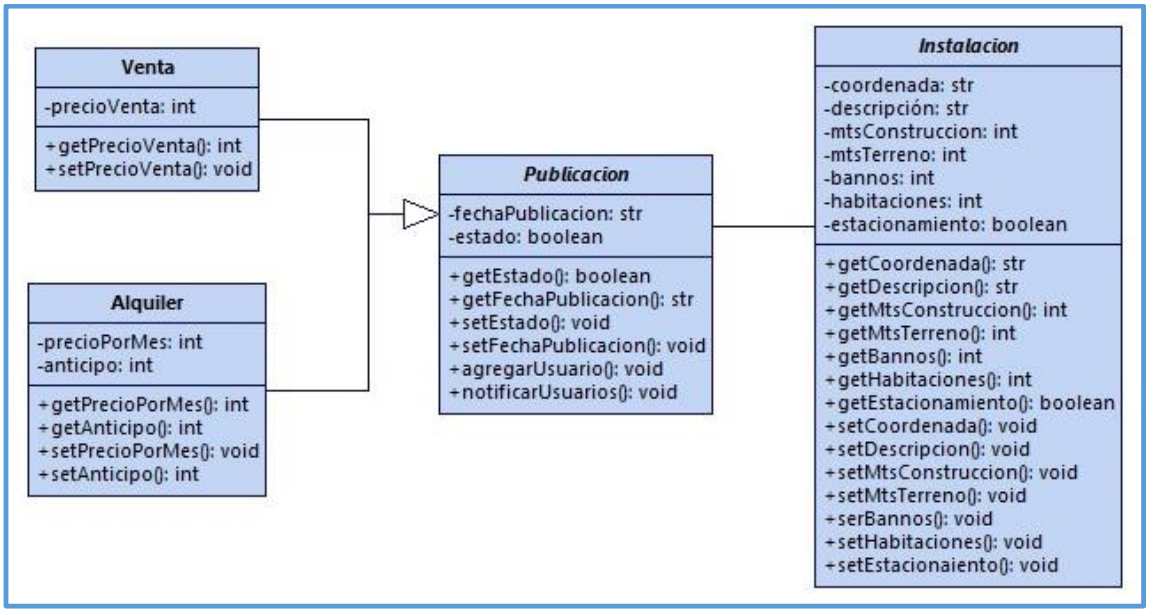


Figura 2 Jugador Rol

6.3. Abstracción Ocurrencia

Como se muestra en la figura 4, la clase edificio se va a componer de una lista de pisos, cada uno de los pisos puede contar ya sea de Apartamentos o de oficinas, dependiendo de la selección que haga el usuario a la hora de crear una instalación, si la instalación a crear es un apartamento u oficina el usuario debe seleccionar un edificio así como un piso del mismo.

El patrón es implementado para facilitar la obtención de datos de un mismo edificio en el mapa, así como guiar al usuario a la hora de la creación de una oficina o apartamento, también ayuda a que no exista repetición de datos.

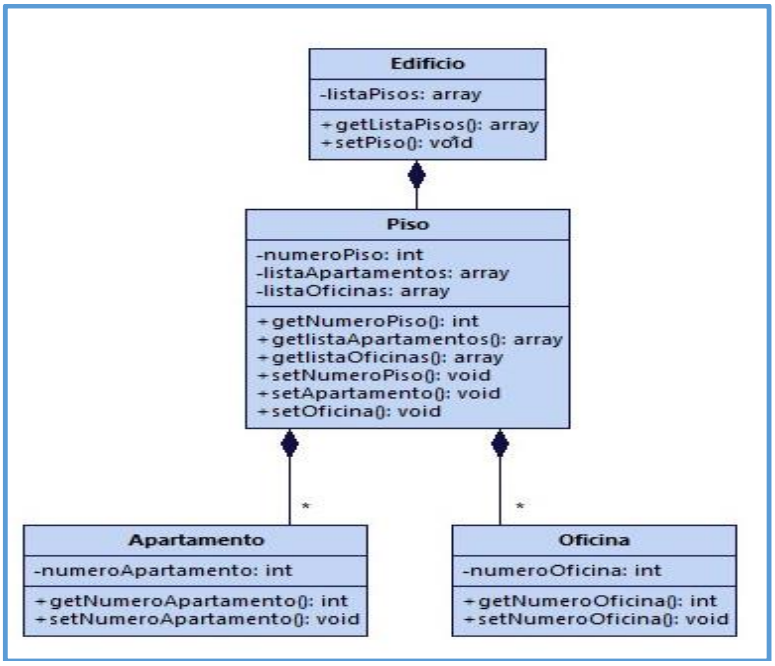


Figura 3 Abstracción Ocurrencia

6.4. Observador Observable

En el momento en que un usuario crea una publicación, va a ser visible para los demás usuarios que estén en línea en ese momento en buscochoza.info, a partir de ese momento el usuario debe estar anuente a cambios en las publicaciones, por lo tanto aquí aplica el patrón observador, estableciendo a una venta o alquiler como un observable concreto y ya sea a un usuario básico o un usuario administrador como el observador concreto.
No afecta en que el observable concreto sea venta o alquiler ya que las dos clases cumplen el mismo papel de ser mostrados a los usuarios. Lo mismo aplica para el observador concreto, sea básico o administrador.

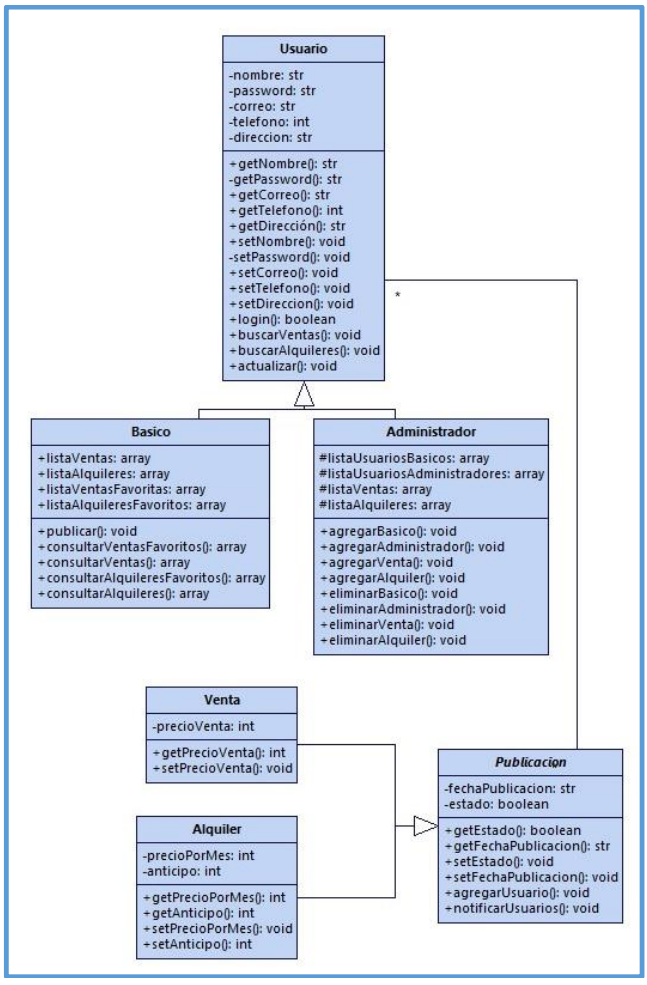


Figura 4 Observador Observable

6.5. Fachada

Se utilizan los servicios web como una fachada ya que se quiere estructurar varios subsistemas en capas y que estos sean los puntos de entrada, de esta manera también se reduce dependencias entre los subsistemas y los clientes.
En la aplicación buscochoza.info se puede ver a la hora de modificar datos a la base de datos ya que no se hace directamente, las entradas pasan por el servicio web para así poder acceder a la lógica donde se realizan las modificaciones o viceversa dependiendo de la función deseada y lo que se quiera hacer con la base de datos, ya sea ingresar nuevos datos, o eliminar datos existentes.

7. Problemas de Diseño

A la hora de empezar a incorporar patrones de diseño en el diagrama de clases se presentaron diversos problemas, esto debido a la falta de conocimiento y práctica de implementación con los patrones de diseño.

7.1. Implementación de patrones de diseño en el diagrama de clases

El principal problema obtenido fue la implementación de los patrones dentro del diagrama de clases, esto debido que fue difícil encontrar los cinco patrones que se solicitaban, por lo que algunos se tu-vieron que forzar.

7.2. Incorporar las clases dentro de un componente en el diagrama de despliegue

Otro problema fue analizar y revisar que todas las clases del diagrama de clases estuvieran mapeadas a un componente definido.

8. Interacción con sistemas externos

Para brindarle más usabilidad al usuario la aplicación deberá conectarse a dos API's externos, los cuales serán:

8.1. Google Maps

La aplicación se conectará con el API de google maps, así se contará con un mapa que mostrará las publicaciones de venta o alquiler realizadas, para que se le facilite a los usuarios la ubicación de las mismas.
Al crear una publicación el usuario tendrá la opción de que su publicación se muestre en el mapa.
El usuario que este en busca de casa o apartamento tendrá varias opciones para afinar su búsqueda en el mapa por medio de búsqueda por provincia, cantón y distrito, por posición actual y por un rango de distancia.

8.2. Facebook

La aplicación se conectará con el API de Facebook, de esta manera el usuario se podrá conectar desde su cuenta de Facebook sin tener la necesidad de crear una nueva cuenta de usuario para la aplicación.
Se tomarán los datos necesarios de la cuenta de Facebook para que el usuario pueda realizar tanto publicaciones como visitar casas o apartamentos de su interés.

9. Otros detalles

La aplicación en un principio tenía el siguiente logo.

9.1. Primer logo

Se actualiza el logo de la aplicación debido a un requisito del interesado.

9.2. Logo Actual

9.3. Dominio

Debido a los costos en los que se debió incurrir por el momento se trabajará con la página http://buscochoza.info/ , más adelante se actualizará a buscochoza.com