|  |  |
| --- | --- |
| Geo Parking | Modelado del Sistema  2014 |

# Control de la documentación

### Control de la Configuración.

|  |  |
| --- | --- |
| Título: | Modelado del Sistema |
| Referencia: | GeoP\_Proyecto\_ModeladoDelSistema.docx |
| Autores: | Lucas Toneatto |
| Fecha: | 30/09/2014 |

### Histórico de Versiones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Estado | Responsable | Cambios |
| 1.0\_DraftA | 30/09/2014 | Pendiente de Revisión | Lucas Toneatto[autor] |  |
| 1.1\_DraftB | 14/10/2014 | Pendiente de Revisión | Lucas Toneatto |  |
| 1.1 | 25/10/2014 | Aprobado | Leonel Romero |  |
| 1.2\_DraftA | 01/02/2015 | Pendiente de Revisión | Lucas Toneatto | Se actualizaron modelos de Negocio, reglas de negocio, Web Service y Presentación. Quedan pendientes cambios en modelos de comportamiento. |
| 1.2\_DrafB | 05/02/2015 | Pendiente de Revisión | Ignacio Frigerio | Se actualizo el diagrama de clases del modelo de negocio |
| 1.0 | 21/02/15 |  | Ezequiel Bär Coch | Actualización y reestructuración de estilos en homogeneización con demás documentos. Se crea tabla de imágenes y sección acrónimos. Se agrega explicación de diagrama de clases móvil. |

Contenido

[Control de la documentación 2](#_Toc412304816)

[Control de la Configuración. 2](#_Toc412304817)

[Histórico de Versiones. 2](#_Toc412304818)

[Acrónimos 5](#_Toc412304819)

[Introducción 6](#_Toc412304820)

[Modelo de negocio 7](#_Toc412304821)

[Modelo de diseño 8](#_Toc412304822)

[Diseño aplicación WEB 9](#_Toc412304823)

[Diseño aplicación MÓVIL 12](#_Toc412304824)

[Modelo de interacción 14](#_Toc412304825)

[Aplicación web 15](#_Toc412304826)

[Búsqueda Playas por Ciudad – Web 15](#_Toc412304827)

[Búsqueda de Playas por Ciudad por Filtro – Web 16](#_Toc412304828)

[Login de Usuario – Web 17](#_Toc412304829)

[Registro de Usuario – Web 18](#_Toc412304830)

[Registro de Playa – Web 19](#_Toc412304831)

[Aplicación Móvil 20](#_Toc412304832)

[Consultar playas – Móvil 20](#_Toc412304833)

[Ir a playa – Móvil 20](#_Toc412304834)

[Enviar consultas para estadísticas – Móvil 20](#_Toc412304835)

[Guardar playa en historial – Móvil 20](#_Toc412304836)

[Buscar playas según un punto de interés – Móvil 21](#_Toc412304837)

[Guardar configuración – Móvil 21](#_Toc412304838)

[Guardar posición de vehículo – Móvil 22](#_Toc412304839)

[Dirigirse a Vehículo – Móvil 23](#_Toc412304840)

[Mostrar listado de playas con disponibilidad - Móvil 24](#_Toc412304841)

[Modelo de arquitectura y despliegue 25](#_Toc412304842)

[Servidor 25](#_Toc412304843)

[ Presentación 25](#_Toc412304844)

[ Negocio 25](#_Toc412304845)

[ Entity Framework 26](#_Toc412304846)

[ SQL Server 26](#_Toc412304847)

[Cliente Web 26](#_Toc412304848)

[Cliente Móvil 26](#_Toc412304849)

[Seguridad 26](#_Toc412304850)

[Conexión con API GeoParking 27](#_Toc412304851)

[Modelo de arquitectura AJAX (Asynchronous JavaScript And XML) 28](#_Toc412304852)

[Modelo de componentes 29](#_Toc412304853)

[Vista de Componentes 29](#_Toc412304854)

[Vista de Datos 30](#_Toc412304855)

[Vista de Negocio 31](#_Toc412304856)

[Vista de Entidades 32](#_Toc412304857)

[Vista de Reglas de Negocio 33](#_Toc412304858)

[Vista de WebService 34](#_Toc412304859)

[Vista de Presentación 35](#_Toc412304860)

[Modelo de patrones 36](#_Toc412304861)

[Singleton 36](#_Toc412304862)

[Repositorio 37](#_Toc412304863)

### Tabla de ilustraciones

[Imagen 1: DC - Del negocio 8](#_Toc412304932)

[Imagen 2: DC - Diseño aplicación web 10](#_Toc412304933)

[Imagen 3: DC - Gestores aplicación web 12](#_Toc412304934)

[Imagen 4: DC - Aplicación móvil 13](#_Toc412304935)

[Imagen 5: DS - Búsqueda de playas por ciudad 16](#_Toc412304936)

[Imagen 6: DS - Búsqueda de playas en ciudad por filtros 17](#_Toc412304937)

[Imagen 7: DS - Inicio y cierre de sesión 18](#_Toc412304938)

[Imagen 8: DS - Registro de usuario 19](#_Toc412304939)

[Imagen 9: DS - Registrar nueva playa 20](#_Toc412304940)

[Imagen 10: DS - Consultar playas móvil 21](#_Toc412304941)

[Imagen 11: DS - Ir a playa, guardar playa en historial y enviar estadísticas 22](#_Toc412304942)

[Imagen 12: DS - Guardar ajustes 23](#_Toc412304943)

[Imagen 13: DS - Guardar posición vehículo 24](#_Toc412304944)

[Imagen 14: DS - Regresar a la posición del vehículo 24](#_Toc412304945)

[Imagen 15: DS - Listado de playas con disponibilidad 25](#_Toc412304946)

[Imagen 16: DD - Modelo de arquitectura 26](file:///Z:\Facultad\Facultad\5°%20AÑO\Proyecto\BackupsRepositorios\GeoParking_0914\GeoParking\Producto\Web\Modelado\GeoP_Producto_Doc_ModeladoDelSistema_1.1.docx#_Toc412304947)

[Imagen 17: DC- Conexión con API GeoParking 28](#_Toc412304948)

[Imagen 18: DD - Vista funcional completa 29](#_Toc412304949)

[Imagen 19: Vista de componentes 30](#_Toc412304950)

[Imagen 20: Vista de datos 31](#_Toc412304951)

[Imagen 21: Diagrama de paquetes de Negocio 32](#_Toc412304952)

[Imagen 22: Vista de entidades 33](#_Toc412304953)

[Imagen 23: Vista de reglas de negocio 34](#_Toc412304954)

[Imagen 24: Vista de Web Service 35](#_Toc412304955)

[Imagen 25: Vista de presentación 36](#_Toc412304956)

[Imagen 26: Patrón Singleton 37](#_Toc412304957)

[Imagen 27: Patrón Repositorio 38](#_Toc412304958)

# Acrónimos

El siguiente listado tiene como objetivo referenciar a los acrónimos utilizados durante el desarrollo del presente documento.

|  |  |
| --- | --- |
| Acrónimo | Significado |
| BD | Base de datos |
| DC | Diagrama de clases |
| DS | Diagrama de secuencia |
| DD | Diagrama de despliegue |
|  |  |
|  |  |

# Introducción

El presente documento técnico constituye parte del desarrollo e implementación del Sistema GeoParking. Lo que se podrá observar es la distribución de los principales componentes que componen el sistema GeoParking.

Describe el modelado del sistema del entorno Web y el entorno Móvil, como parte complementaria de la Arquitectura del Sistema. El modelado incluye tres perspectivas para describir el sistema, las cuales son:

* El Modelo de Negocio
* El Modelo de Arquitectura
* El Modelo de Comportamiento

En síntesis el presente documento, comprende la estructura del sistema GeoParking, donde el diseño de la solución considera desde el ingreso de datos e información, la manera en que estos estarán almacenados, así como la forma en que estos se procesarán, analizarán y presentarán al usuario final.

Las imágenes que se mostrarán a continuación son diagramas de modelos que tratan de esquematizar la forma en que se estructura y se comporta el sistema GeoParking.

# Modelo de negocio

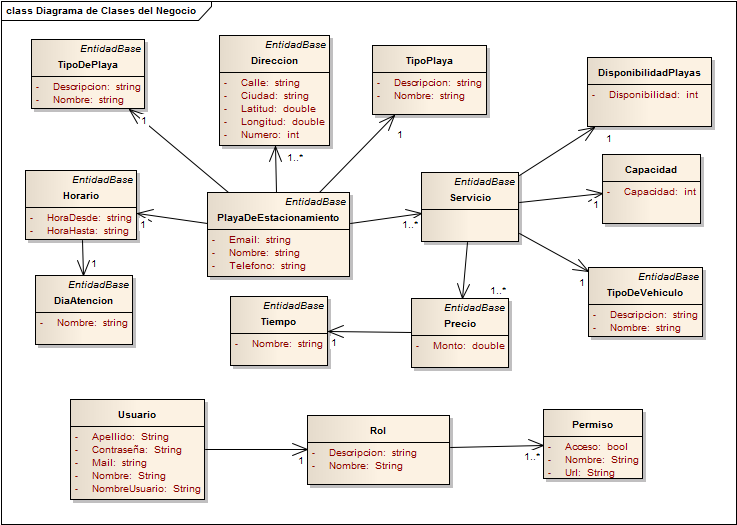


Imagen 1: DC - Del negocio

Como se puede observar el modelo de negocio esta dirigido a abarcar todos aquellos elementos y características que se destacan en el negocio de playas de estacionamiento, como ser: playas, vehículos, tipos de vehículos, horarios, días de atención, precios, disponibilidades, etc.

El elemento principal de este negocio es la **Playa de Estacionamiento** y de ella es que derivan los demás conformando sus características principales. El dominio del negocio que se plantea en el modelo es el de estacionamiento de vehículos particulares en playas de estacionamiento. Es hacia ese dominio, que se inserta GeoParking para dar solución a problemas y generar mejoras en el manejo de información correspondiente.

Y por lo general, en la mayoría de los sistemas, el modelo de negocio también abarca al usuario, el cual interactúa con esos elementos donde su rol es brindar datos u obtener información a partir de los mismos.

# Modelo de diseño

A continuación se mostrará la estructura principal del diseño que da soporte a la solución implementada. Se podrá visualizar tanto el diseño para la aplicación web como para la aplicación móvil.

## Diseño aplicación WEB

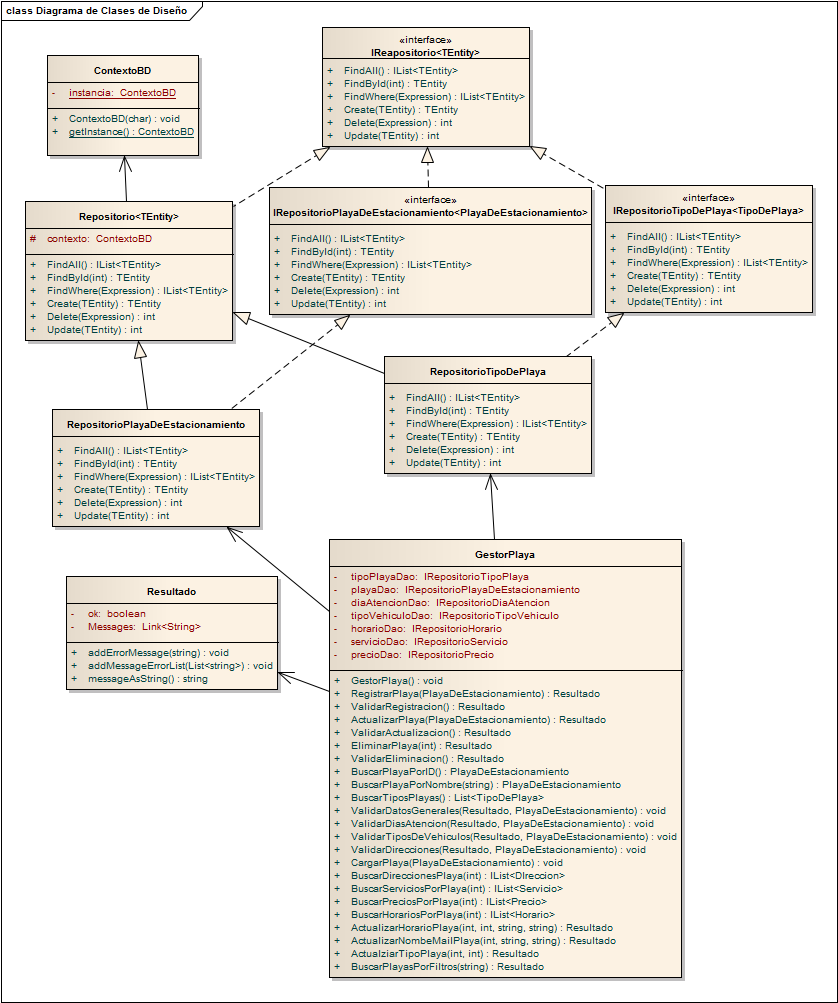


Imagen 2: DC - Diseño aplicación web

En la imagen se puede apreciar el modelo de solución implementado por GeoParking, en el cual se destaca la implementación de un patrón llamado **REPOSITORIO**, utilizado para dar soporte a la recuperación de información desde la BD y como esta es mapeada en entidades a nivel de dominio.

Todo esto se puede lograr través de la intervención de **GESTORES**, especializados en manejar un grupo de entidades específicas.

Estos **GESTORES** se dedican a manejar las entidades y realizar con ellas operaciones sobre la BD. Operaciones tales como búsqueda, inserción, eliminación y consultas.

A modo de simplificar el diagrama, solo se puede visualizar a modo de ejemplo el GestorPlaya, pero los demás gestores, se presentan en la solución con las mismas relaciones destacadas en este diagrama.

A continuación se ilustran los demás gestores que conforman el diseño de la solución GeoParking.

**Nota:** Para una mayor visualización del contenido de la imagen, remitirse al documento digital anexado.

**

Imagen 3: DC - Gestores aplicación web

## Diseño aplicación MÓVIL



Imagen 4: DC - Aplicación móvil

Como se muestra en la imagen anterior, el funcionamiento se centra en un elemento html llamado **index.html** el cual representa la capa de presentación al usuario y por detrás existe un conjunto de operaciones (reglas de negocio) en distintos archivos confeccionados para tal fin:

El archivo **Utils.js** es de propósito general facilitando operaciones como inicializar mensajes, ejecutar conversiones de sistema decimal a radian para la manipulación del mapa, validar formularios, entre otros. Tiene como atributo el **db** que representa el objeto **localStorage** con el cual se hace uso de la base de datos interna del dispositivo.

Se cuenta con un conjunto de widgets:

* **cargarMapa.js:** este es el que inicia el mapa utilizando la API de **Google Maps v3.**
* **obtenerPlayas.js:** se encarga de conectarse con el servidor mediante peticiones AJAX al WebService, para obtener las playas registradas en la base de datos y mostrarlas en el mapa.
* **historialWidget.js:** el cual tiene como objetivo es almacenar y recuperar la información sobre las últimas playas consultadas por el usuario móvil, almacenándose con la tecnología **localStorage**, en el propio dispositivo móvil.
* **preferencesWidget.js:** su principal objetivo es almacenar y recuperar la información brindada por el usuario sobre el tipo de vehículo seleccionado, el radio de distancia de búsqueda y la activación/desactivación del GPS.
* **contactoWidget.js:** como su nombre indica, tiene como función principal, crear un formulario web y realizar petición AJAX al WebService para el envío de correos.
* **Autocomplete.**js: realiza la inicialización del objeto places de la librería Places de la API v3 de google maps, con el objetivo de buscar y obtener un punto de interés en el mapa.

**Nota:** Para una mayor visualización del contenido de la imagen, remitirse al documento digital anexado.

# Modelo de interacción

En los siguientes diagramas de secuencia lo que se trata de mostrar es la interacción general de como los componentes del sistema interactúan entre sí para poder brindar la funcionalidad que se tiene como objetivo.

El objetivo de estos diagramas es brindar una trazabilidad a nivel de código que permita a simple vista observar, captar y analizar el impacto del cambio de alguno de los componentes intervinientes en cada funcionalidad, permitiéndole todo esto, obtener una visión más precisa del impacto de los cambios a realizar.

## Aplicación web

### Búsqueda Playas por Ciudad – Web

Aquí el usuario web introduce una ciudad y se realiza una búsqueda de las playas de estacionamiento que existan en esa ciudad.

La petición se realiza en la página “**web.aspx**” (página de inicio de búsqueda del sistema). Esta última envía a la página encargada de las búsquedas, la ciudad que ingreso el usuario, y la página “**BusquedaPlaya.aspx**” es la que se encarga de comenzar la búsqueda.

Se conecta con el gestor encargado de manejar las búsquedas de playas de estacionamiento y le envía la ciudad donde quiere recuperar las playas.

Como último paso el gestor retorna las playas resultantes de la búsqueda hacia la página de visualización para el usuario.

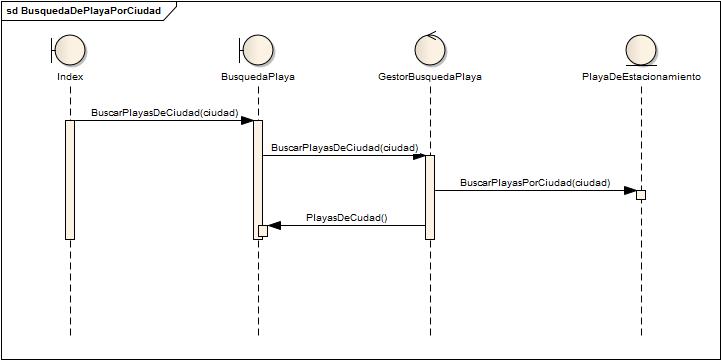


Imagen 5: DS - Búsqueda de playas por ciudad

### Búsqueda de Playas por Ciudad por Filtro – Web

Aquí el usuario web selecciona una serie de filtros que están disponibles y se realiza una búsqueda de las playas de estacionamiento que cumplan con esas características.

La petición se realiza en la página “**BusquedaPlaya**.aspx”. Esta última se conecta con el gestor encargado de manejar las búsquedas de playas de estacionamiento y le envía los parámetros (filtros) para realizar la búsqueda de las playas que concuerden con ese criterio.

Como último paso el gestor retorna las playas resultantes de la búsqueda hacia la página de visualización para el usuario.

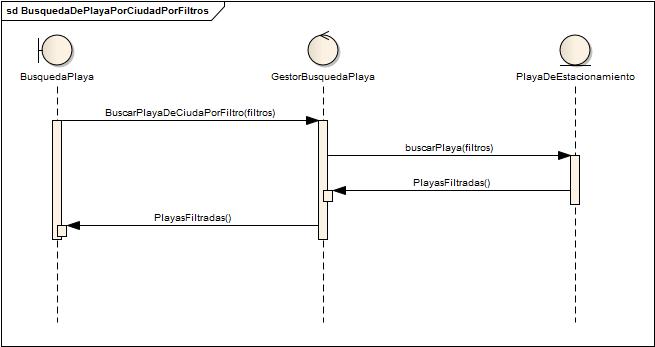


Imagen 6: DS - Búsqueda de playas en ciudad por filtros

### Login de Usuario – Web

Inicio de Sesión: Aquí el usuario web selecciona la opción de ingresar al sistema, y se le muestra una ventana donde deberá ingresar usuario y contraseña correspondientes. Una vez enviada la petición de logeo el sistema primero validara que los datos requeridos (usuario y contraseña) estén completos y con el formato correcto.

Una vez validados en la página, se envían dichos datos al servidor para que sean validados y se inicie la nueva sesión del usuario si es que los datos aportados fueron validados correctamente. Caso contrario, el sistema emitirá un mensaje de error, notificando el motivo del mismo.

Cierre de Sesión: Aquí el usuario web selecciona la opción cerrar sesión y el sistema toma los datos de la sesión actual y elimina del sistema la misma.

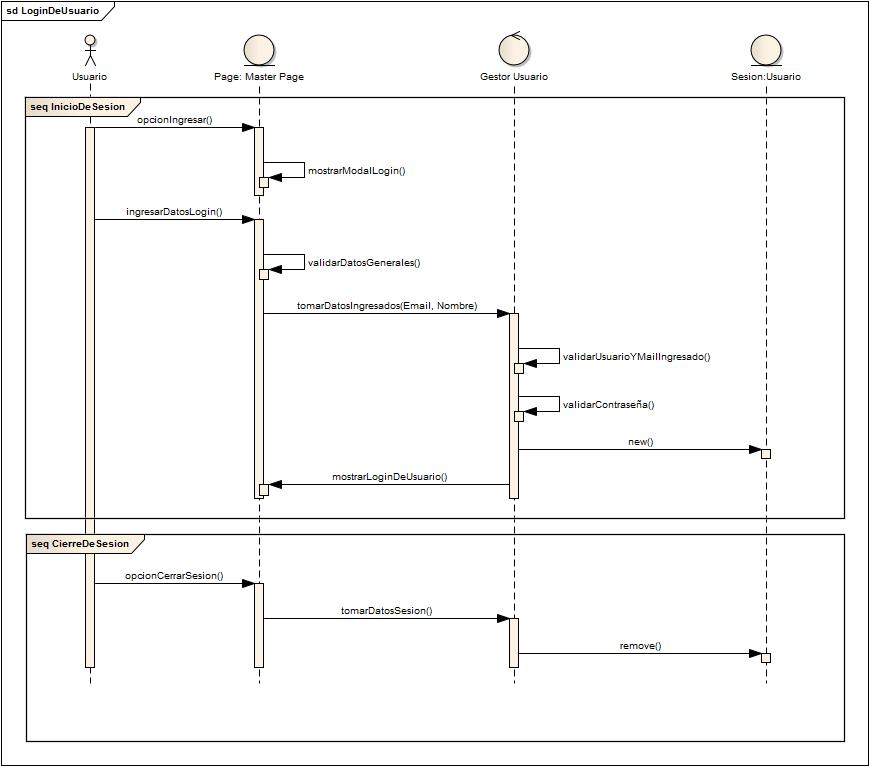


Imagen 7: DS - Inicio y cierre de sesión

### Registro de Usuario – Web

Aquí el usuario desea ingresar al sistema, y como no posee datos de acceso, selecciona la opción de registrarse. En ese momento se le muestra una ventana donde deberá ingresar los datos para la registración. Se realiza una validación de que los datos solicitados estén completos y en el formato correcto.

Una vez validados, se envía los datos al servidor donde se realiza una nueva validación, en este caso ya de consistencia con la información ya almacenada en el sistema. Un ejemplo de esos controles es de usuario existente en el sistema.

Con el control realizado y validado, se procede la creación del nuevo usuario en el sistema y se emite un mensaje de registración exitosa. Caso contrario, se emite un mensaje de error notificando el motivo del mismo.

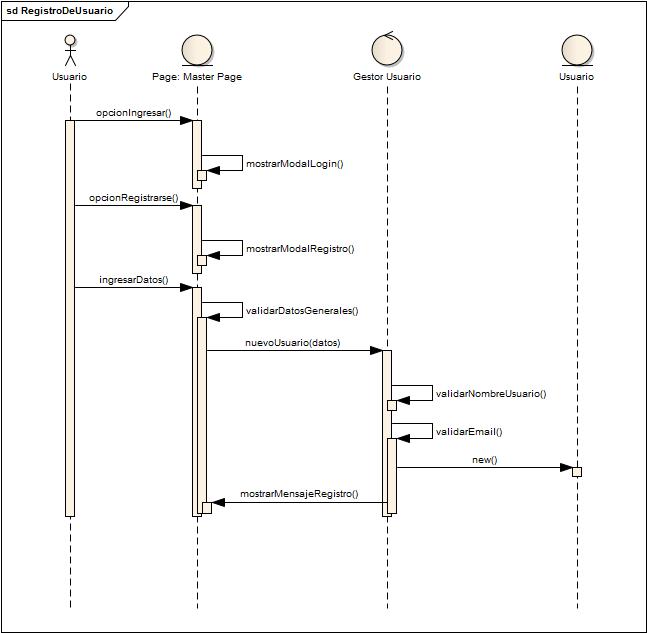


Imagen 8: DS - Registro de usuario

### Registro de Playa – Web

Aquí el usuario desea ingresar una nueva playa al sistema y por ello selecciona la opción “Nueva Playa”. En ese momento se le mostrara una ventana donde se le solicitan todos los datos necesarios de la playa de estacionamiento para su registración.

Antes de enviar los datos al servidor, se realiza una validación en el cliente donde se evalúa que los datos estén completos, en el formato correcto y consistente entre ellos.

Una vez validados en el cliente se envían los datos al servidor para un nuevo control de los mismos a nivel de sistema y si estos son correctos, se registra la playa de estacionamiento y se muestra el mensaje de registración exitosa. Caso contrario, el sistema emitirá un mensaje de error, notificando el motivo del mismo.

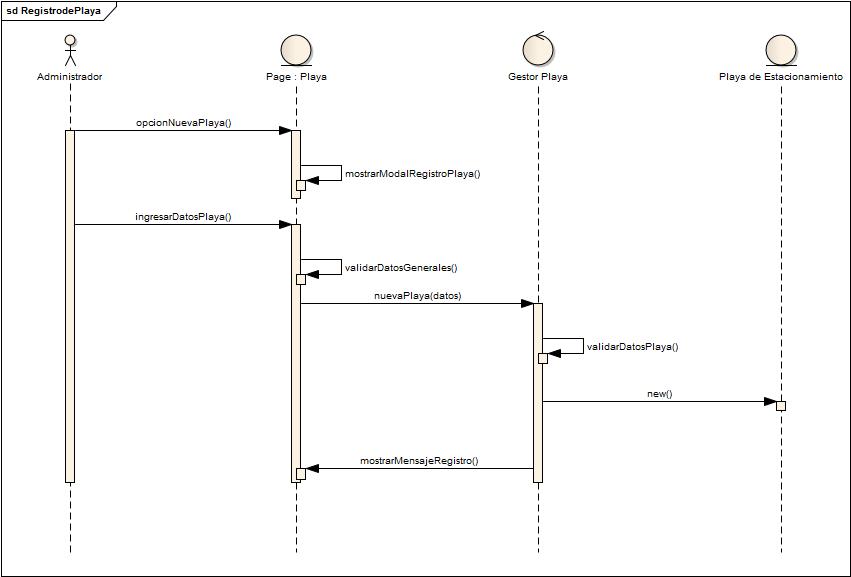


Imagen 9: DS - Registrar nueva playa

## Aplicación Móvil

### Consultar playas – Móvil

Al inicializarse la aplicación se recurre a la consulta de las playas de estacionamiento. Para ello se obtiene la posición actual del usuario (su dispositivo) y se realiza la llamada al web service para buscar las playas de estacionamiento que existan en la posición pasada por parámetro y que cumpla con el filtro de tipo de vehículos, definido por el usuario durante la configuración de los ajustes. Luego estas playas son mostradas en el mapa mediante un marcador.



Imagen 10: DS - Consultar playas móvil

### Ir a playa – Móvil

El usuario podrá visualizar el trayecto desde su posición actual hacia la playa seleccionada. Para ello, se toma la posición actual del usuario (dispositivo móvil), y el usuario podrá optar por presionar sobre un marcador de playa con su respectivo icono o desde el listado de playas generado en el apartado anterior una playa, lo que provocará que el sistema trace el recorrido por la ruta más corta posible hacia el objetivo final. Además, el usuario podrá visualizar la trayectoria presionando el botón “mostrarTrayectoria”. Ver **imagen 11**.

### Enviar consultas para estadísticas – Móvil

Tiene como objetivo alimentar al módulo de estadísticas enviando datos acerca de la ubicación desde donde se realiza la consulta, el tipo de vehículo utilizado, las playas consultadas, etc.

Para ello, cada vez que el usuario consulta una playa ya sea desde el listado de playas disponibles explicadas en el inciso anterior, se envían los datos y se almacenan a través del web service. Luego estos, serán procesados y reflejados en los informes estadísticos explicados más adelante. Ver **imagen 11**.

### Guardar playa en historial – Móvil

El usuario podrá tener almacenadas un conjunto de playas que son las que él consulto, ordenadas por antigüedad de consulta. Es decir, las playas más recientes aparecerán al inicio del listado mientras que las consultadas con más anterioridad serán las que ocupen la base de la lista. Ver **imagen 11**.



Imagen 11: DS - Ir a playa, guardar playa en historial y enviar estadísticas

### Buscar playas según un punto de interés – Móvil

Con el fin de proporcionarle al usuario una forma de búsqueda a través de lugares donde pueda tener interés dirigirse o buscar lugar de estacionamiento. Estos podrían ser shoppings, parques, centros de salud, entes gubernamentales, universidades, etc.

Para ello el usuario ingresa un punto de interés o dirección, y el sistema responderá si encuentra o no dicha solicitud. En caso afirmativo le permitirá mostrar lo solicitado en el mapa y podrá buscar playas de estacionamiento cercanas en el radio comprendido entre el punto de interés y el radio configurado de búsqueda.

### Guardar configuración – Móvil

El objetivo es almacenar las preferencias del usuario en cuanto al tipo de vehículo que utiliza, el radio de búsqueda de playas y la opción de habilitar /deshabilitar el GPS. Todo esto se almacena en el dispositivo en la **localStorage.** Esto es válido tanto en la pantalla de bienvenida (cuando es la primera vez que utiliza la aplicación GeoParking en el dispositivo) como en la pantalla de ajustes, donde puede configurar cuando el usuario lo requiera.

****

Imagen 12: DS - Guardar ajustes

### Guardar posición de vehículo – Móvil

El usuario podrá almacenar la ubicación donde deje estacionado su vehículo, ya sea esta en una playa de estacionamiento o en un lugar disponible en la vía pública. Esto le permitirá luego, poder visualizar la trayectoria desde el lugar en donde se encuentra (mediante su dispositivo móvil) hasta el vehículo en cuestión. Este proceso se explica en el siguiente apartado: **Dirigirse a vehículo**.



Imagen 13: DS - Guardar posición vehículo

### Dirigirse a Vehículo – Móvil

Como se mencionó anteriormente, el usuario podrá visualizar el trayecto comprendido desde su ubicación actual (dispositivo móvil) hasta la posición donde registró el vehículo con anterioridad. Este trayecto está configurado para ser trazado desde el rol de un peatón.



Imagen 14: DS - Regresar a la posición del vehículo

### Mostrar listado de playas con disponibilidad - Móvil

Para la obtención de playas cercanas al usuario, se toma la ubicación actual del usuario (su dispositivo) a través de geo posicionamiento. Esto, junto a los datos configurados por el usuario en las preferencias (tipo de vehículo, radio de búsqueda y GPS) son enviados al web service cuyo método se encargará de retornar un listado de playas que cumplan con los filtros enviados como parámetros. Esto se presentará al usuario como un listado, mostrando en una escala de colores según su disponibilidad, nombre de las playas, dirección, precios por hora (si los tuviera) y distancia en metros desde la ubicación del usuario hacia la playa.

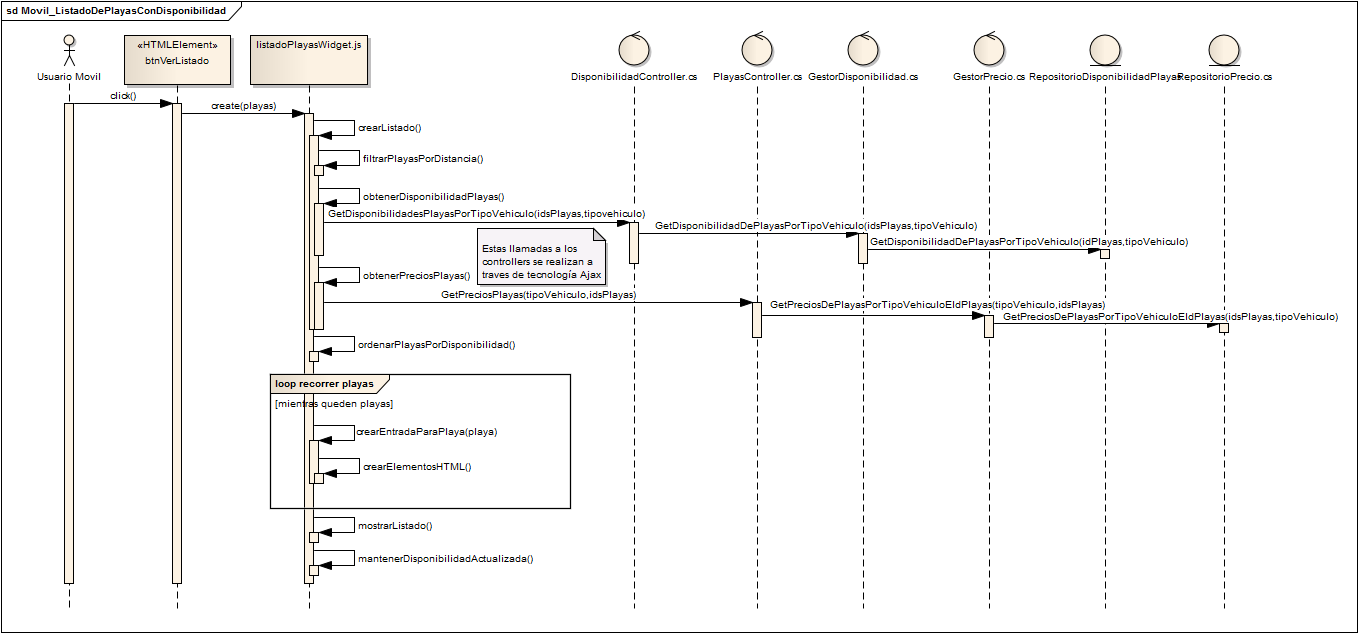
****

Imagen 15: DS - Listado de playas con disponibilidad

**Nota:** Para una mayor visualización del contenido de las imágenes, remitirse al documento digital anexado.

# Modelo de arquitectura y despliegue

**Nota:** Para una mayor visualización del contenido de las imágenes, remitirse al documento digital anexado.

La arquitectura que presenta el sistema GeoParking es como el diagrama que se muestra a continuación. La arquitectura se organiza en 3 bloques principales que están en contante interacción: El servidor de la aplicación, el cliente Web y el Cliente Móvil.

## C:\Users\Lukas\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Modelo de arquitectura.jpgServidor

Imagen 16: DD - Modelo de arquitectura

En este nodo se despliegan 4 componentes, los cuales están encargados de realizar o brindar distintas funciones, es por ello que la arquitectura a desplegar en el servidor está organizada en capas. Las capas son las siguientes:

* Presentación: es el componente encargado de albergar todo aquello que genere una forma de interacción con el usuario, será aquella interfaz de comunicación a través de la cual se ofrezca las funcionalidades del sistema.
* Negocio: es el componente encargado de albergar todo aquello que establezca la manera en que funcionara el sistema, y las reglas que rigen ese comportamiento. Este componente se subdivide a su vez en:
  + **Entidades**: contendrá todos aquellos elementos que formen parte del problema que trata el sistema como a su vez también elementos para la solución de los mismos. Dichos elementos forman parte del modelo del negocio como del modelo de solución.
  + **Reglas de Negocio**: contendrá todo aquellos agentes que regulan el funcionamiento del sistema, y maneja el mundo de entidades que lo conforman.
  + **Web Service**: será el componente que brinde los servicios del sistema a la aplicación móvil, la cual está desarrollada de forma externa al sistema.
* Entity Framework: es aquel componente encargado del mapeo Objeto-Relacional de las entidades del sistema.
* SQL Server: es la instancia de la Base de Datos que alberga toda la información del sistema.

## Cliente Web

Este nodo contendrá el **navegador** a través del cual se realiza la conexión con nuestro sistema mediante de peticiones Http (Internet).

## Cliente Móvil

Este nodo contendrá la **aplicación móvil** desarrollada para que consuma los servicios del Web Service deployado en el Servidor y así podrán brindar las funcionalidades del sistema.

## Seguridad

El manejo de la seguridad de la aplicación en cuanto al acceso de la información que el sistema obtiene, procesa y muestra, está basado en la autenticación de quien se comunica con el sistema ya sea web o desde la aplicación móvil.

En cuanto a la aplicación web la seguridad de acceso a la información esta manejada por el manejo de roles y permisos para los diferentes usuarios que interactúan con el sistema. Cada funcionalidad está restringida a roles específicos.

En la aplicación móvil, la obtención de la información que se muestra en la aplicación es obtenida a través de los servicios del componente WebService del sistema. Todas las peticiones que realiza nuestra aplicación móvil envían un clave al servicio web para que valide que es nuestra aplicación la que está realizando esas peticiones y no otra ajena al sistema.

### Conexión con API GeoParking



Imagen 17: DC- Conexión con API GeoParking

Como se puede apreciar los componentes presentes en el modelo, corresponden a los distintos pasos para el uso del API GeoParking por parte de los sistemas de las playas de estacionamiento.

El proceso consta de crear una solicitud, que la misma sea aprobada, y que paso seguido se cree la conexión para comunicar la playa con GeoParking.

Para ello lo destacable es la funcionalidad que proporciona cada uno de los gestores intervinientes en el proceso.

**GestorSolicitud**: es el encargado de crear las solicitudes para una nueva conexión y de actualizar el estado de las mismas.

**GestorConexion**: es el encargado de crear la conexión, que le permite a la playa obtener los datos de acceso al API de GeoParking.

**GestorAutenticacion**: es el encargado de validar la conexión de quien se trata de comunicar con el API de GeoParking y de darle acceso a la modificación de información de las playas.

# Modelo de arquitectura AJAX (Asynchronous JavaScript And XML)

**Nota:** Para una mayor visualización del contenido de las imágenes, remitirse al documento digital anexado.

Aquí se trata de mostrar de qué manera se hace uso de la tecnología AJAX en la implementación de nuestro proyecto GeoParking. En el diagrama se puede visualizar los componentes intervinientes en el flujo de datos.

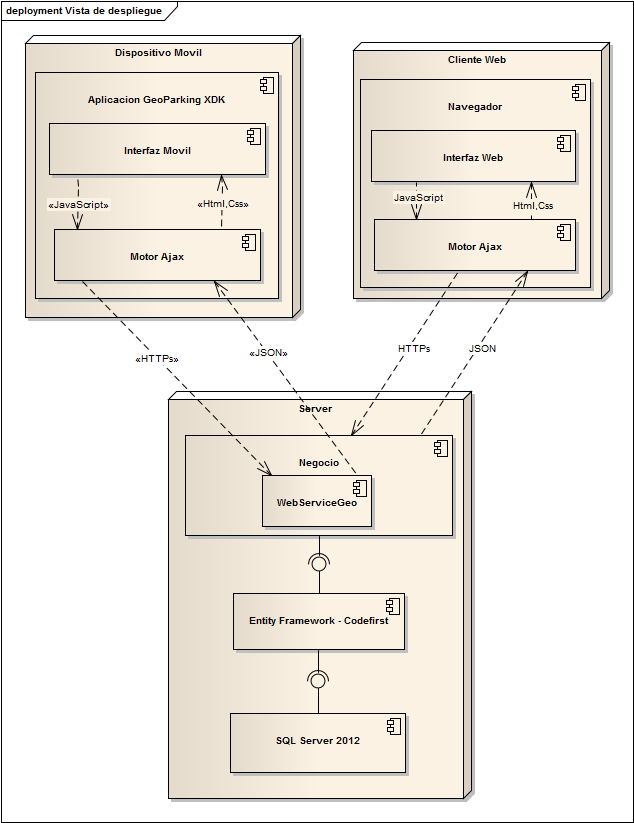


Imagen 18: DD - Vista funcional completa

# Modelo de componentes

**Nota:** Para una mayor visualización del contenido de las imágenes, remitirse al documento digital anexado.

## Vista de Componentes

Aquí se puede observar como en nuestro sistema el componente principal está constituido o estructurado en una división por capas, las cuales son:

1. **Datos**: capa dedicada a la materialización y desmaterialización de los datos de la aplicación, es lo que normalmente se denomina mapeo objeto relacional.
2. **Negocio**: capa dedicada a aplicar la lógica del negocio en el sistema y todas aquellas reglas que rigen su funcionamiento.
3. **Presentación**: capa dedicada a brindar una interfaz visual con la cual los usuarios interactúen con el sistema.

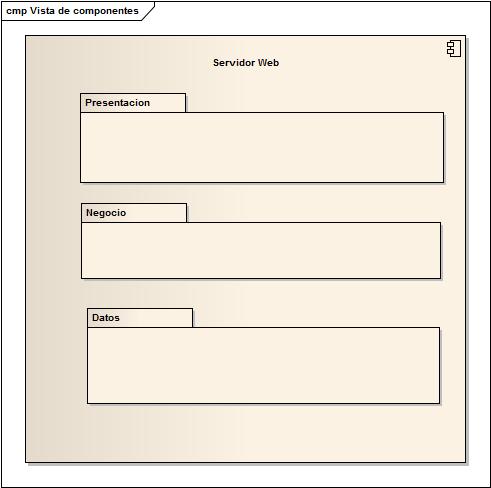


Imagen 19: Vista de componentes

## Vista de Datos

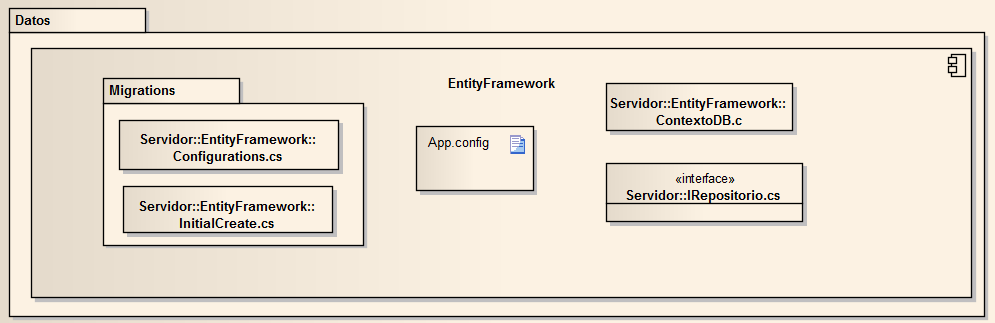


Imagen 20: Vista de datos

Aquí se puede observar como esta capa está basada en la tecnología Entity Framework, empleando el modelo Codefirst. Para ello, estos son los principales componentes que permiten en funcionamiento de esta capa del sistema:

* **Migrations**: este paquete contiene dos clases muy importantes en el funcionamiento de Entity Framework
  + **Configuration.cs**: tiene establecido el contexto de la BD donde se montara la conexión para el mapeo de los objetos.
  + **InitialCreate.cs**: tiene establecido la creación del modelo en memoria de los objetos para el posterior mapeo.
* **App.config**: tiene establecido la conexión a la instancia de la Base de Datos. Y además la versión de Framework con la que debe realizar el mapeo objeto relacional.
* **ContextoBD.cs**: crea el contexto en memoria en relación a los objetos presentes en la Base de Datos. Establece contexto particulares para cada objeto del sistema.
* **IRepositorio:** interfaz que establece las acciones que se pueden realizar sobre cada objeto del sistema en cada contexto.

## Vista de Negocio

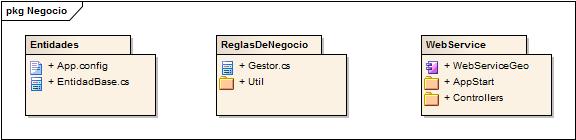


Imagen 21: Diagrama de paquetes de Negocio

Aquí se puede observar como esta capa está estructurada y subdividida en tres paquetes, los cual tienen funcionalidades muy diferentes pero que a su vez se relacionan para cumplir un objetivo en común, brindar la funcionalidad necesaria al usuario aplicando las distintas reglas del negocio. En esta capa se divide en:

1. Entidades
2. ReglasDeNegocio
3. WebService

**Entidades:** alberga todas las clases que representan los objetos manejados en el modelo de negocio de la aplicación.

**ReglasDeNegocio:** alberga todos los controladores que aplican las distintas reglas de negocio sobre los objetos/entidades del sistema.

**WebService:** alberga la aplicación que se corresponde al webService, el cual permite ofrecer la funcionalidad del sistema a la aplicación móvil.

## Vista de Entidades

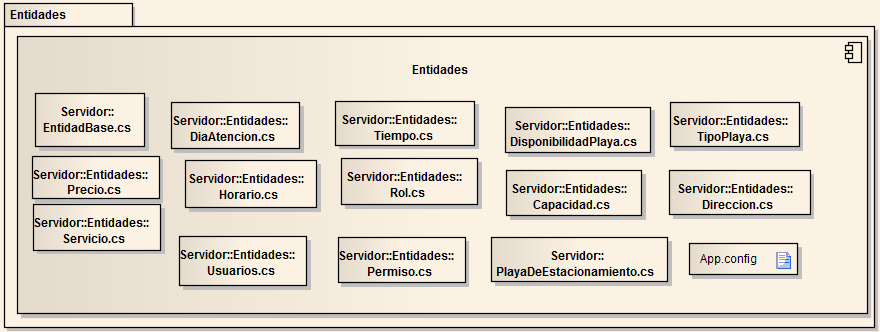


Imagen 22: Vista de entidades

Como se aprecia este componente alberga todas las entidades que dan soporte al modelo de negocio sobre el que trabaja el sistema GeoParking. Contiene los siguientes ítems:

* **Clase Base**: la clase EntidadBase.cs es la clase madre de todas las demás clases del negocio, de ella heredan las clases hijas.
* **Clases Hijas**: son todas aquellas clases que heredan de EntidadBase.css y ellas son:
  + **PlayaDeEstacionamiento.cs**
  + **TipoPlaya.cs**
  + **Serivicio.cs**
  + **Precio.cs**
  + **DiaAtencion.cs**
  + **Tiempo.cs**
  + **Horario.cs**
  + **Direccion.cs**
  + **Provincia.cs**
  + **Departamento.cs**
  + **Ciudad.cs**
  + **Usuario.cs**
  + **Rol.cs**
  + **Permiso.cs**
* **App.config:** archivo de configuración del componente.

## Vista de Reglas de Negocio

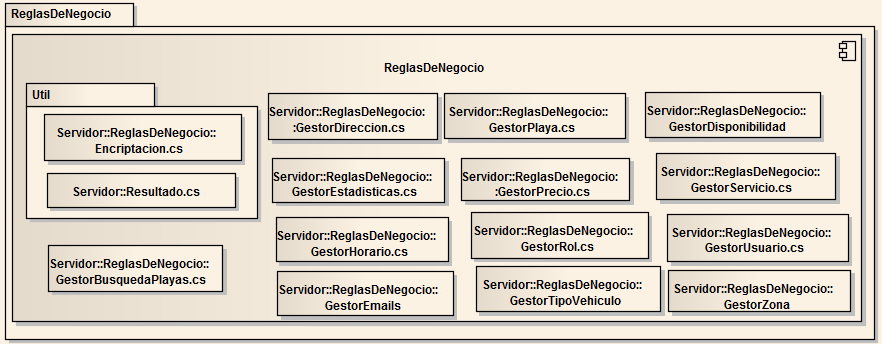


Imagen 23: Vista de reglas de negocio

Esta vista lo que muestra es como están soportadas las reglas de negocio del sistema (comportamiento), a través de componentes de software. En el componente principal que hace referencia al módulo de negocio se puede distinguir dos tipos de componentes importantes:

* **Paquete Útil**: aquí se albergan dos clases importantes en el manejo del sistema. Una de ellas es la clase Encriptación.cs, la cual maneja la seguridad de la información del usuario con su sesión activa. Y la otra clase es Resultado.cs, la cual permite retornar de manera más estructurada mensajes alusivos a los resultados del procesamiento que la información sufre en esta capa. Toda acción que se realice en esta capa y que deba retornar un resultado de la misma se hace a través de esta clase.
* **Gestores**: son las clases encargadas del procesamiento de las peticiones que se realizan al sistema. Cada gestor está encargado de una funcionalidad específica del sistema.

## Vista de WebService

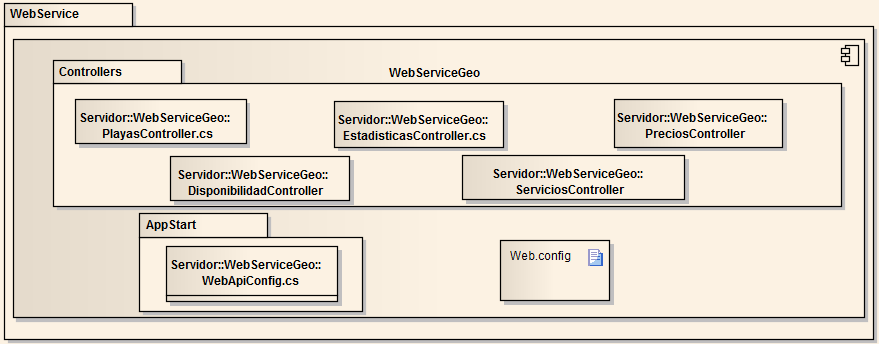


Imagen 24: Vista de Web Service

Aquí se puede observar que el componente principal es el servicio web WebServiceGeo, el cual interactúa con los demás componentes de la capa de negocio para que la funcionalidad que provee el sistema esté disponible a través de la aplicación móvil. Este servicio web se compone de:

* **WebApiConfig.cs:** establece el formato de ruta con la cual se puede acceder a través de un protocolo de internet a las distintas funcionalidades que ofrece el WebService.
* **Web.config:** establece la conexión con la BD.
* **Paquete Controllers:** este alberga todos los controladores que realizan procesamientos de información por cada una de las consultas que le llegan al web Service. Los controladores definidos en el web Service son:
  + **PlayasController.cs**: controlador que brinda la funcionalidad de manejar todas aquellas peticiones relacionadas al manejo de información de las playas de estacionamiento del sistema. Se encarga de buscar playas y retornarlas a la aplicación que las requiera, de acuerdo a los criterios recibidos en la petición.
  + **EstadisticasController.cs**: controlador que brinda la funcionalidad de manejar todas aquellas peticiones relacionadas al manejo de información estadística del sistema relacionada a la aplicación Móvil. Se encarga de tomar todos los datos que la aplicación móvil pueda brindarnos para luego realizar las estadísticas e informes correspondientes.

## Vista de Presentación

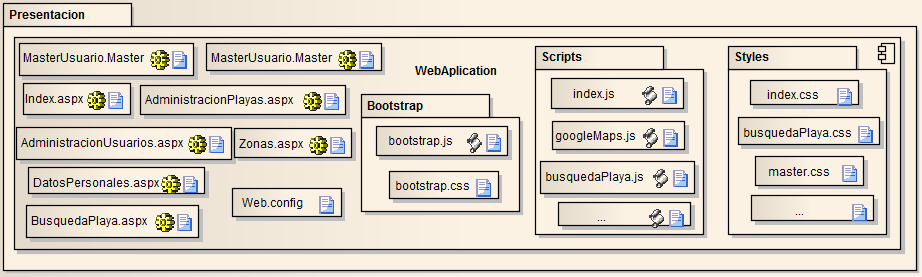


Imagen 25: Vista de presentación

Aquí se puede observar como esta capa del sistema está conformada por una aplicación web donde se pueden observar componentes como:

* **Site.master:** platilla modelo de interfaz de donde heredan las demás páginas.
* **Paginas.aspx**: páginas que heredan de la master. Las paginas desarrolladas hasta el momento son:
  + **Index**.aspx
  + **Ayuda**.aspx
  + **Contacto**.aspx
  + **Playa**.aspx
  + **AdministracionUsuarios**.aspx
  + **BusquedaPlaya**.aspx
  + **Faq**.aspx
* **Web.config**: configuración de distintos aspectos de la aplicación web.
* **Paquete Bootstrap**: contiene todos los archivos JavaScript y Css (estilos) del framework Bootstrap (librería de terceros) para el maquetado de interfaces web.
* **Paquete Scripts**: contiene todos los archivos JavaScript de creación propia aplicados a la interactividad del usuario con componentes de la interfaz web del sistema.
* **Paquete Styles**: contiene todos los archivos de estilos (.Css) de creación propia, que actúan en el diseño de la interfaz visual del sistema, modificando su apariencia.

# Modelo de patrones

Aquí se muestra el diagrama de implementación de los patrones utilizados en el desarrollo del producto GeoParking.

## Singleton

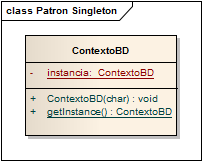


Imagen 26: Patrón Singleton

**JUSTIFICACION**

La aplicación del patrón Singleton sobre la clase ContextoBD surge por:

* Necesidad de que exista una instancia única en todo el sistema; esto es porque a la hora de instanciarse un objeto del tipo ContextoBD requiere de un tiempo para levantar toda la estructura para relacionarse con la base de datos, y poder mapear todos los objetos. Si se instanciara cada vez que se lo requiere la performance de la aplicación decaería precipitosamente.
* Acceso desde todas las clases DAO: por cada clase entidad, necesitaremos una clase específica que realice todas las tareas relacionadas a la base de datos a través de las funcionalidades que le preste el contexto.

## Repositorio

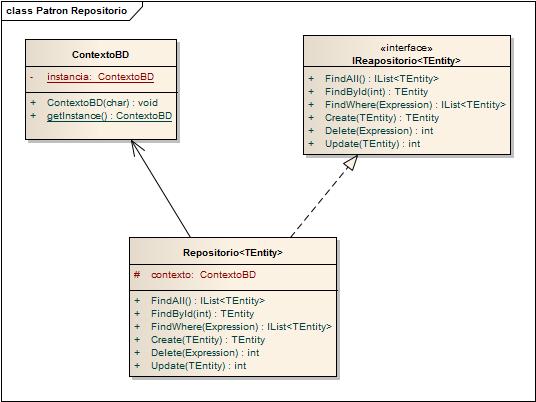


Imagen 27: Patrón Repositorio

**JUSTIFICACIÓN**

* En pocas palabras, un repositorio es un mediador entre el dominio de la aplicación y los datos que le dan persistencia. Con este planteamiento podemos pensar que el usuario de este repositorio no necesitaría conocer la tecnología utilizada para acceder a los datos, sino que le bastaría con saber las operaciones que nos facilita este “mediador”, el repositorio.
* Utilizando como tecnología de acceso a datos el ORM Entity Framework, con una aproximación “code-first”, se podrá utilizar el repositorio para cualquier entidad compleja que se defina.