Turismo Móvil

Documentación técnica

Materia: Desarrollo de Software

Docente: Ernesto Mayorga

Alumno: Enzo Scalone

Establecimiento: Universidad Nacional de la Patagonia San Juan Bosco – Facultad de Ingeniería.

Año: 2020

Índice

Tecnologías utilizadas

Plataforma web

Lenguaje de programación utilizados:

- PHP 7.2
- Javascript

Base de datos

- MySql 5.7

ORM

- Eloquent (laravel)

Frameworks y librerías principales

- Laravel 5.2
- AngularJS 1.7.2
- Bootstrap 3.3.7

Servidor web

- Apache HTTP Server

Versionado de código

- GIT
- Repositorios provistos por bitbucket.org

Servicios externos

- TodoPago (deprecado) para cobros con tarjetas de crédito

Aplicación Móvil

Lenguaje de programación utilizados:

- Java (Android) nativo.

Base de datos

- SQLite nativo

Frameworks y librerías principales

- Retrofit 2.4: Llamadas HTTP a la API web
- Dagger 2.17: Inyección de dependencias
- Glide 4.8: Administración multimedia (imágenes)
- Joda-Time 2.10: Librería para el trabajo con fechas.

IDE

- Android Studio

Versionado de código

- GIT
- Repositorios provistos por bitbucket.org

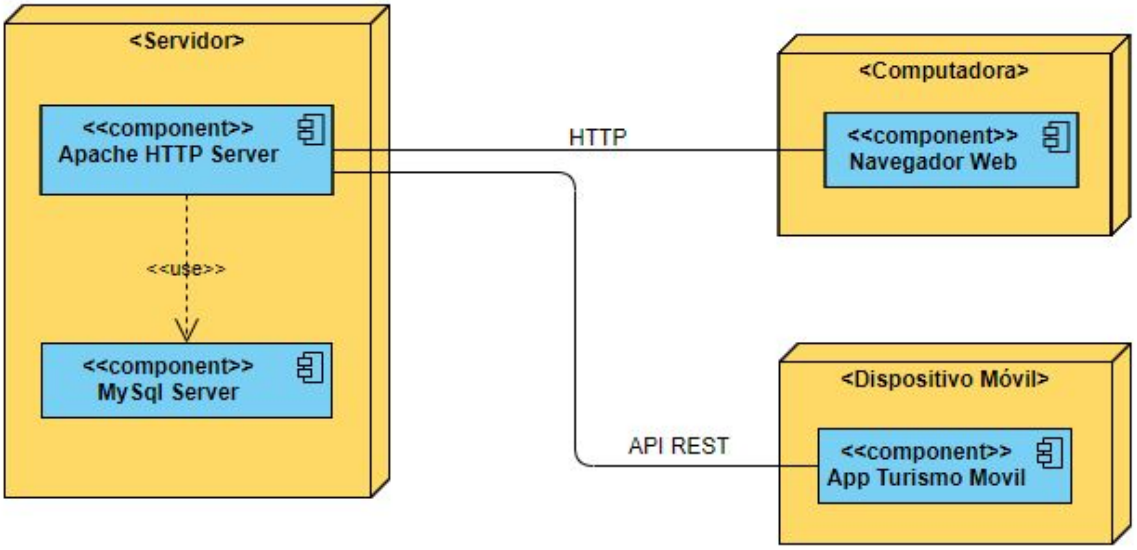
Servicios externos

- OpenWeatherMap: API para recibir el pronóstico

Arquitectura del sistema web

El sistema sigue una arquitectura web simple. Provee mediante una API rest, los servicios a los distintos dispositivos móviles utilizados por los turistas. Se utiliza el formato JSON para la transmisión de datos.

Los usuarios del sistema web (administradores y clientes) utilizan un navegador estándar, ya que se trata de una plataforma web.



Estructura de la base de datos

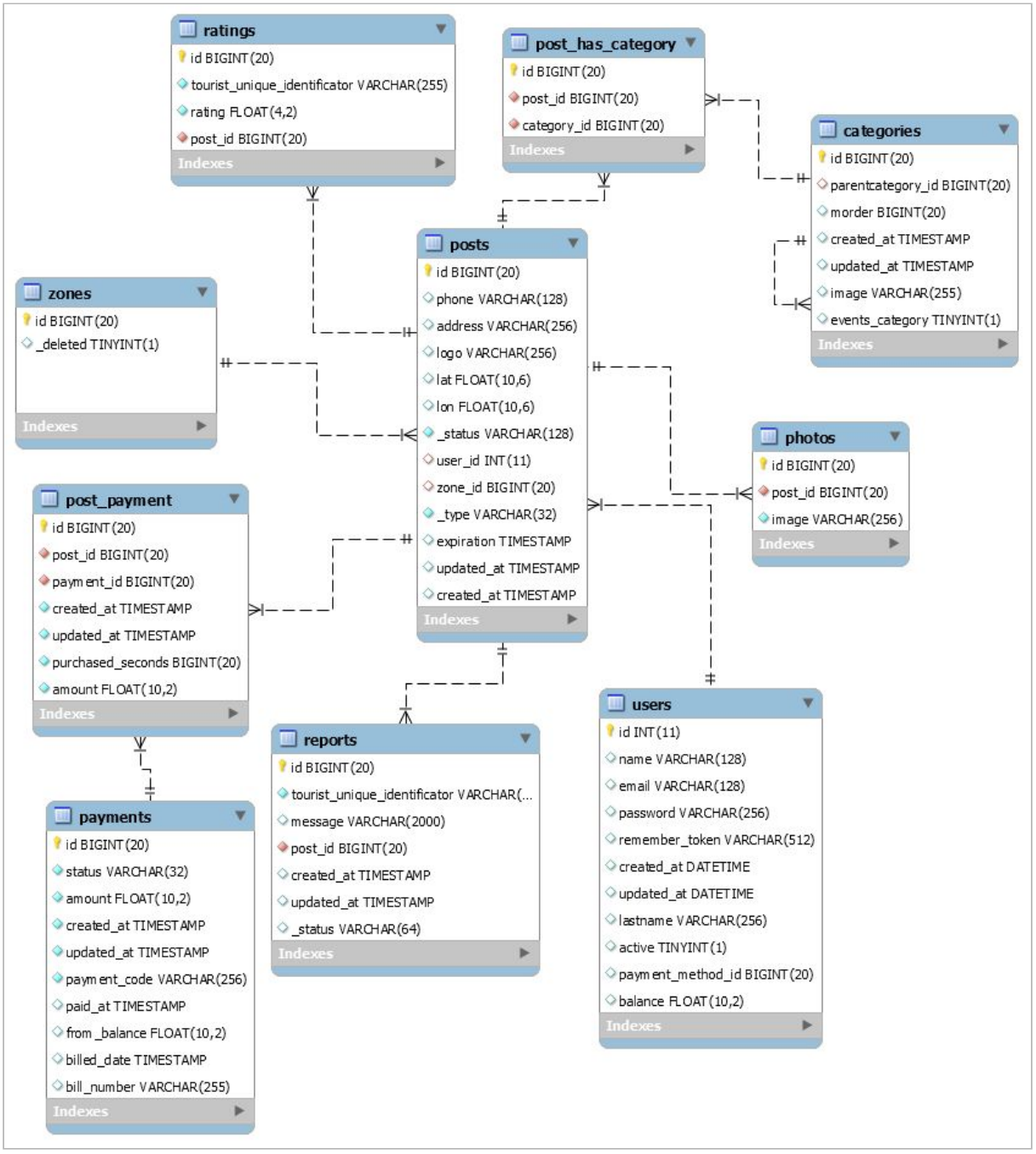
En los siguientes diagramas se mostrará el diseño de la base de datos utilizada en el sistema.

Considerando el número de tablas existentes, se simplificó el diagrama en cada caso ilustrando únicamente las tablas relacionadas con la funcionalidad indicada.

Publicaciones

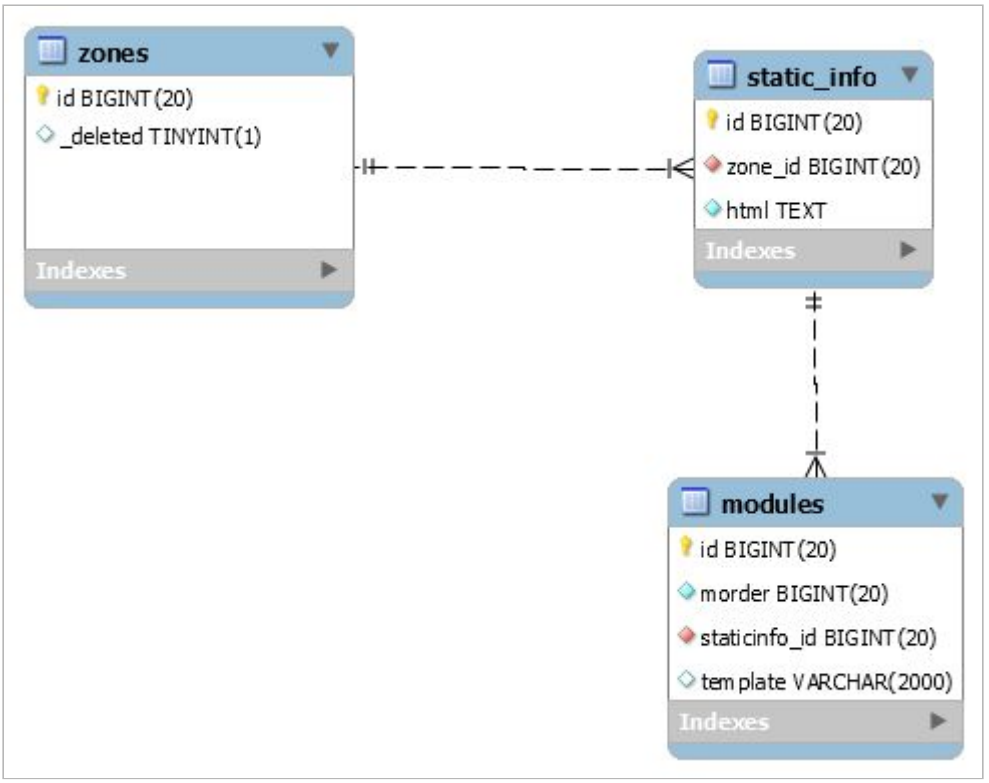
Se puede observar un detalle en el siguiente diagrama, que los elementos que requieren internacionalización, no están representados en las tablas, como por ejemplo, el nombre y la descripción de los “posts” o el nombre de las categorías. Esos atributos se contemplan en un sistema de traducción genérico.

Respecto al pago de las publicaciones, se diseñó de modo que quede identificado mediante la tabla “post_payment”, cuanto tiempo se pagó, y cual fué el valor de la promoción en cada caso en particular. Esto se realizó para permitir hacer una devolución del tiempo restante respecto a la promoción pagada, en caso de eliminar una publicación, o cambiar su tipo.



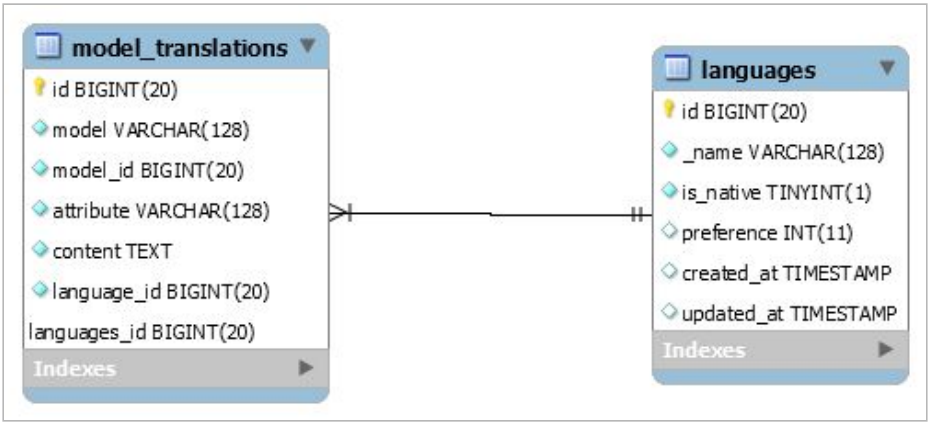
Información estática

Con la información estática (o información regional) ocurre lo mismo que las publicaciones respecto a los elementos que requieren traducción. No se almacenan directamente en la tabla.
Respecto a los módulos, se puede ver que posee un orden “morder” y una plantilla “template”. La plantilla indica la disposición de los datos y el tipo de módulo que es, pero no el contenido en sí, ya que debe ser traducible.



Tablas de internacionalización

Para realizar la traducción de los atributos internacionalizables de cualquier tabla, se utilizó un enfoque genérico.



Cada texto que requiere traducción pertenece a un atributo de una tabla (o de un modelo, visto desde el ORM de laravel), por lo que se implementó una tabla donde se contemplan estos elementos:

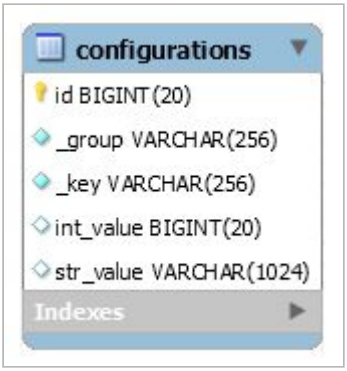
- **model**: Es el nombre de la clase en el ORM de laravel. Por ejemplo “Post” o “Category”
- **model_id**: El id numérico de la entidad.
- **attribute**: El nombre de la columna que se está traduciendo
- **content**: La traducción en sí
- **language_id**: El lenguaje asociado a la traducción

De esta forma se puede traducir cualquier elemento del sistema sin generar tablas específicas.
Ejemplo:

model	model_id	attribute	content	language_id
Category	158	name	Natural Foods	1
Category	5	name	Hotel	2
Post	82	description	No te comas el verso. Veni a Mostaza	2
Post	116	_name	Evento piola	2
StaticInfo	12	title	Informacion de Madryn	2
Category	8	name	temp	1
Post	101	description	Eventos!	2

Configuraciones

Se implementó una tabla que relaciona un valor a una clave, de forma que se puedan guardar configuraciones.



- **_group**: Grupo que engloba un conjunto de configuraciones
- **_key**: La clave por la cual se accede al valor
- **int_value** y **str_value**: El valor en sí de la configuración, se contemplaron valores numéricos y en forma de texto.

Por ejemplo, en caso de la configuración de cantidad de categorías por tipo de publicación, la tabla se vería así:

_group	_key	int_value	str_value
cat_amount_by_post_type	event	99	*NULL*
cat_amount_by_post_type	free	1	*NULL*
cat_amount_by_post_type	silver	2	*NULL*
cat_amount_by_post_type	gold	3	*NULL*

Estructura del proyecto

A continuación se indican las rutas de mayor interés para el desarrollo del proyecto.

Ruta	Descripción
app/Http/Controllers	Los controladores que responden a las rutas de la API, separados por funcionalidad.
app/Http/Services	Servicios de la aplicación. Actualmente solo se utilizó para separar la funcionalidad de TodoPago.
app/	En la raíz de la carpeta app se encuentran además de los paquetes indicados anteriormente, los modelos del ORM Eloquent.
app/Http/routes.php	Rutas de la API
front_end/	Se encuentran los paquetes de funcionalidad del front end, es decir, la aplicación angular que se ejecuta en el navegador

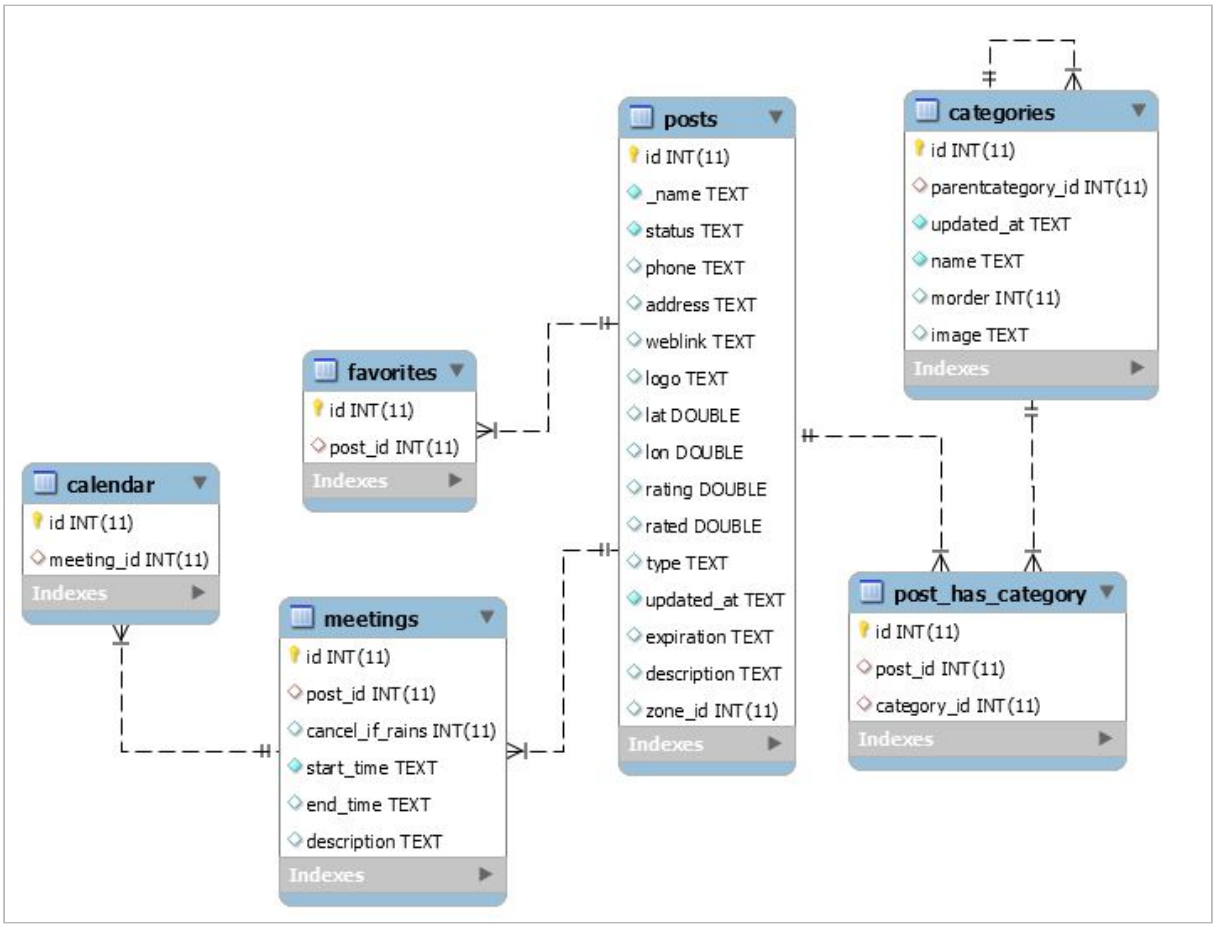
Arquitectura de la aplicación móvil

La aplicación móvil se desarrolló utilizando el entorno de desarrollo de Android nativo. Se utilizó Android Studio como IDE, que utiliza Gradle para la instalación de dependencias.

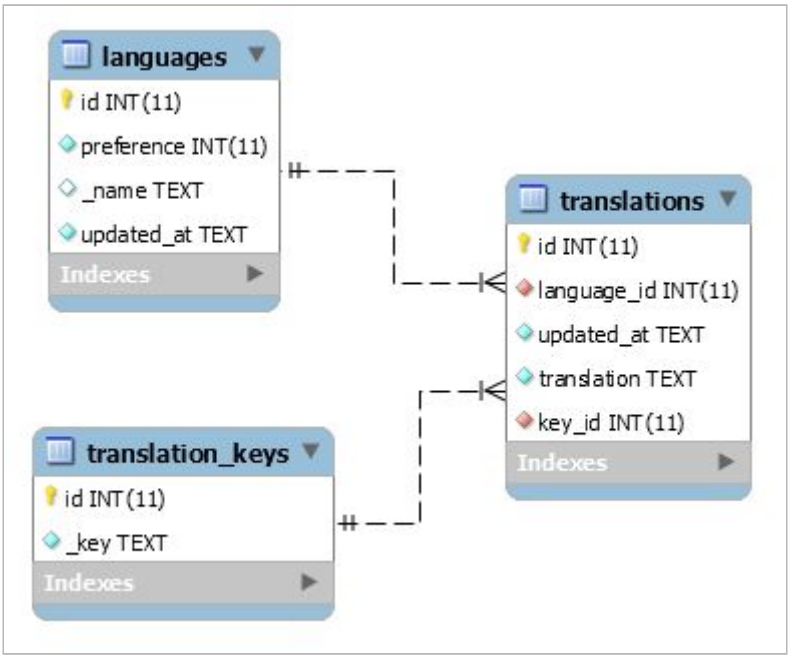
Estructura de la base de datos

La base de datos contemplada en la aplicación móvil es reducida, ya que no se contemplan todos los elementos del sistema web. Por ejemplo, en las publicaciones, se mantiene solo la traducción del lenguaje seleccionado, y no todas las traducciones. En el caso de seleccionar otro lenguaje, se debe descargar toda la nueva traducción de cada publicación.

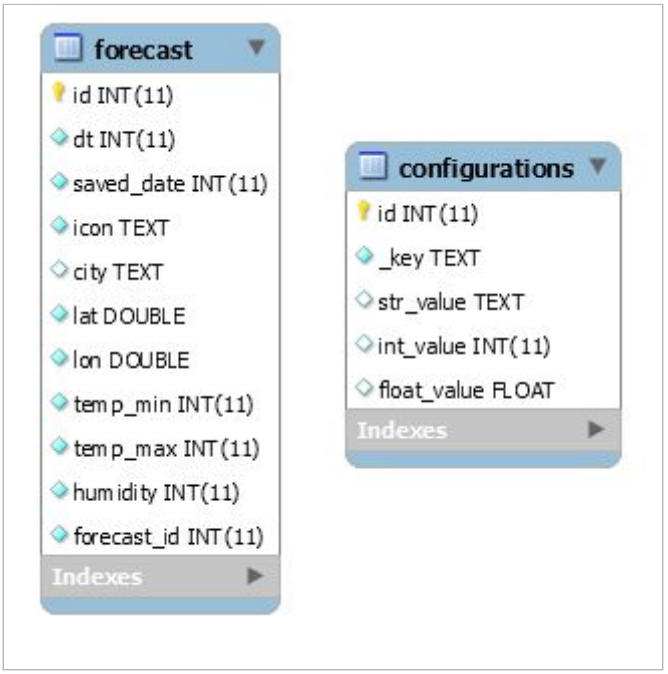
Publicaciones



Traducción de interfaz



Tablas de configuración y pronóstico



Estructura del proyecto

A diferencia del proyecto web, donde existe un framework que fuerza la disposición de los paquetes, en Android el programador tiene que elegir e implementar un estructura de proyecto.

Paquete	Descripción
turismomovil.adapters	Adaptadores de datos de las listas (ListViews)
turismomovil.application	Definición de clases base de la Aplicación y Activity
turismomovil.daos	Interfaces de los Datos
turismomovil.daos.impl	Implementaciones de los Datos
turismomovil.di	Paquete con las configuraciones de Dagger (inyección de dependencias). Se encuentra la configuración inicial del sistema junto con la definición de la base de datos
turismomovil.dialogs	Diálogos personalizados
turismomovil.dto	Data transfer objects (Objetos de transferencia de datos) Utilizado principalmente para realizar la conversión del JSON recibido desde la API
turismomovil.exceptions	Excepciones personalizadas
turismomovil.http	Servicios HTTP en formato requerido para trabajar con Retrofit
turismomovil.models	Modelos utilizado por los Daos
turismomovil.services	Servicios de lógica de negocios
turismomovil.utils	Clases estáticas con funciones útiles como el formato de fechas o cálculo de kilómetros.
turismomovil.* (raíz)	Activities de android

Configuraciones esenciales

Antes de realizar las pruebas en la aplicación móvil, se debe configurar la dirección del servidor. Este valor se encuentra en el siguiente archivo:

`res/values/strings/strings.xml`

Debe cambiar el siguiente elemento
`<string name="server_ip">**http://192.168.1.73:82/**</string>`
Con la dirección IP o el servidor web correspondiente.