LA RUEDA:

Optimización de viajes por trabajo

Anexo 5: Documentación técnica.

Trabajo de Fin de Máster

Máster Universitario en Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

SEPTIEMBRE 2021

AUTOR:

Marcos Unzueta Puente

TUTOR:

Pablo Chamoso Santos

Índice

Índice de ilustraciones

1. Introducción

En este anexo se procede a recoger toda la información sobre la propia programación del sistema software.

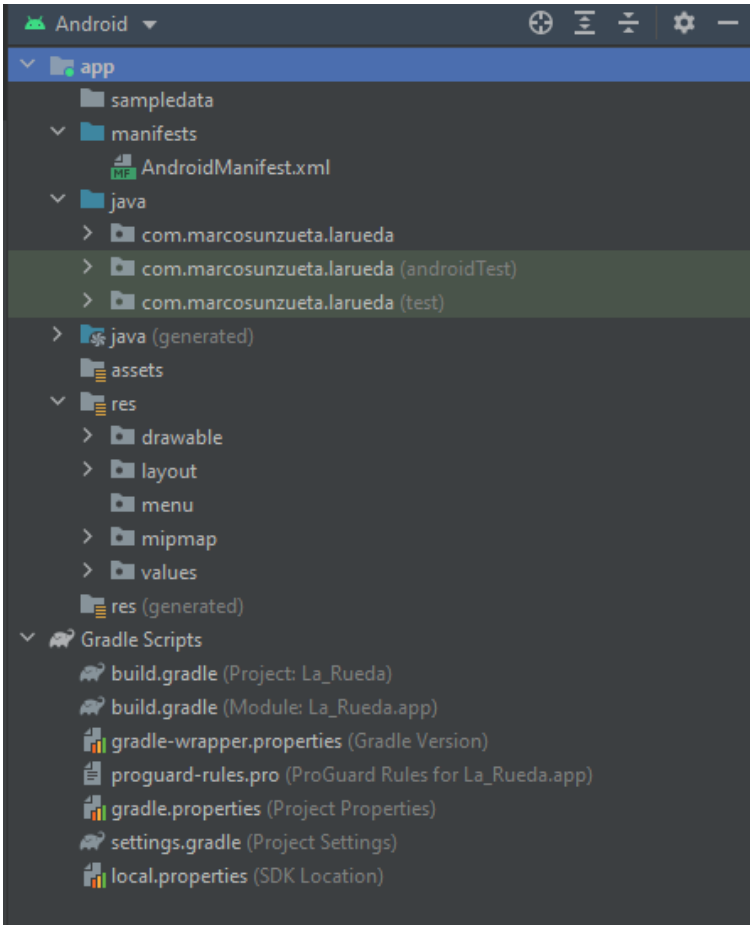
Se mostrará la estructura de directorios generada en el espacio de trabajo de Android Studio, así como los archivos que se almacenan en estos.

2. Documentación técnica de la aplicación

El producto software generado para la realización de este Trabajo de Fin de Máster se ha llevado a cabo haciendo uso del entorno de desarrollo integrado Android Studio, por tanto, la distribución de directorios y ficheros estará condicionada por la estructura por defecto existente en el propio entorno mencionado.

El espacio de trabajo utilizado tendrá la siguiente estructura:

Tabla 1: Esquema de la aplicación



Se procederá a explicar los elementos más importantes haciendo uso de Javadoc siempre que sea posible:

2.1. Manifests

La carpeta de manifests únicamente contiene el fichero AndroidManifest.xml

Tabla 2: Esquema de manifests

2.1.1. AndroidManifest.xml

Este fichero contiene la información esencial de la aplicación para las herramientas de creación de Android, el sistema operativo Android y Google Play.

Utiliza XML para definir, entre otras cosas, las actividades de la aplicación y su estilo, algunos datos de configuración, los permisos que la aplicación necesita del usuario y las direcciones.

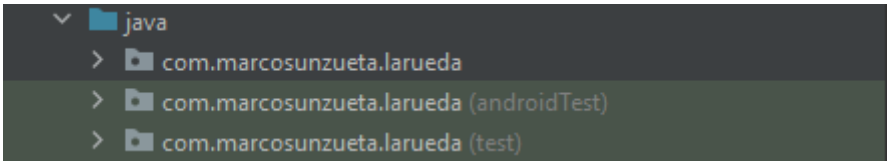
En esta aplicación, en concreto, ha sido necesario darle permisos para que pueda escribir y leer en el almacenamiento externo del dispositivo móvil. (Android, s.f.)

2.2. Java

En esta carpeta se almacenará el código fuente desarrollado para la aplicación.

El backend de la aplicación está formado por archivos de java que siguen el patrón de arquitectura modelo-vista-controlador, así como clases auxiliares que permiten llevar a cabo la gestión de la base de datos y la reutilización de funciones y datos entre otras cosas.

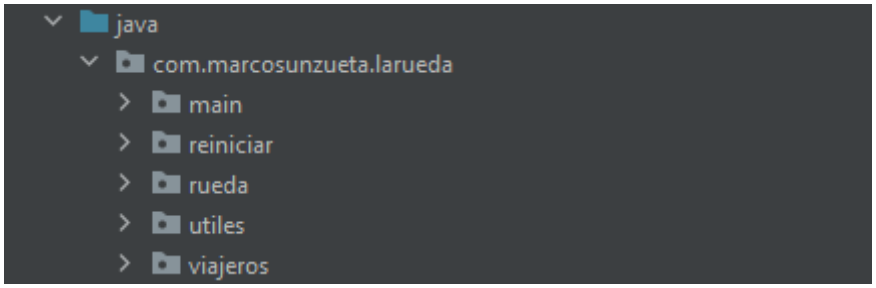
Tabla 3: Esquema de java



Los ficheros de java se encuentran distribuidos en carpetas en la dirección com.nombredesarrollador.nombreproyecto, siendo en este caso com.marcosunzueta.larueda el nombre del directorio.

En esta memoria, se desarrollarán aquellos ficheros, clases, métodos o atributos que resulten de interés y se omitirá la información que pueda resultar muy simple o redundante.

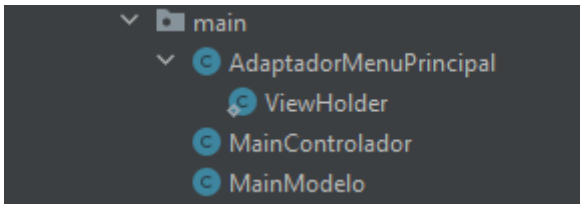
Tabla 4: Esquema de java.com.marcosunzueta.larueda



2.2.1. Main

En la carpeta Main se encuentran los ficheros que se encargan del funcionamiento del menú principal de la aplicación.

Tabla 5: Esquema de java.com.marcosunzueta.larueda.main



2.2.1.1. *AdaptadorMenuPrincipal*

/**

* Aadaptador que se utilizará para dar formato al recycler que muestra a los viajeros almacenados en la aplicación.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **List<Integer> imagenes:** Imágenes que se mostrarán en el menú.
- **List<String> textos:** Textos que se mostrarán en el menú.
- **List<String> botones:** Botones que se mostrarán en el menú.
- **public AdaptadorMenuPrincipal(Context contexto, MainModelo modelo):**

  /**
  * Constructor del adaptador correspondiente al recycler del menú principal.
  *
  * @param contexto Controlador que va a mostrar el menú.
  * @param modelo Modelo que contiene los datos necesarios para desplegar el menú.
  */

2.2.1.2. *AdaptadorMenuPrincipal.ViewHolder*

/**

* ViewHolder que se encargará de gestionar cada elemento de la lista.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.1.3. *MainControlador*

/**

* Controlador del menú principal.

* Muestra las distintas opciones de la aplicación.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **modelo MainModelo:** Modelo correspondiente al controlador.

- **private void ruedaReiniciada():**

  /**

  * Función que mostrará un mensaje al usuario indicándole que la rueda se ha reiniciado.

  */

- **private void ruedaNoReiniciada():**

  /**

  * Función que mostrará un mensaje al usuario indicándole que la rueda no se ha reiniciado.

  */

2.2.1.4. MainModelo

/**

* Modelo del menú principal.

* Muestra las distintas opciones de la aplicación.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **MainControlador controlador:** Controlador del modelo.

- **long momentoDeUltimoClick**: Almacenará el momento en el que se ha realizado click en un botón y prevendrá que el usuario pueda clickar dos veces seguidas.

- **public MainModelo(MainControlador controlador):**

  /**

  * Constructor que sirve para instanicar el modelo.

  *

  * @param controlador Controlador correspondiente al modelo a instanciar.

*/

- void transicionARueda():

  /**

  * Función que realiza una transición a la actividad de rueda.

  */

- **void transicionAViajeros():**

  /**

  * Función que realiza una transición a la actividad de viajeros.

  */

- **void transicionAReiniciar():**

  /**

  * Función que realiza una transición a la actividad de viajeros.

  */

- **void inicializarAlmacenamientoInterno():**

  /**

  * Comprueba si el almacenamiento interno está inicializado, si no es así lo inicializa.

  * El almacenamiento interno únicamente almacenará el id que se irá asignando a los nuevos usuarios,

  * y solo será necesario inicializarlo en caso de que sea la primera vez que se ejecuta la aplciación en el dispositivo.

  */

- **public void reiniciarRueda():**

  /**

  * Función encargada de reiniciar la rueda.

  */

- **void eliminarViajeros():**

  /**

  * Función que eliminará a todos los viajeros de la base de datos y reiniciará los IDs.

  */

- **void eliminarRueda():**

  /**

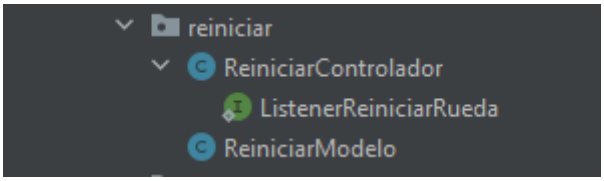
  * Función que eliminará el PDF de la rueda almacenado por la app.

  */

2.2.2. Reiniciar

En la carpeta Reiniciar se encuentran los ficheros que se encargan de resetear los datos de la aplicación .

Tabla 6: Esquema de java.com.marcosunzueta.larueda.reiniciar



2.2.2.1. ReiniciarControlador:

/**

* Controlador del modal correspondiente a reiniciar la aplicación.

* Muestra un modal que permitirá al usuario decidir si desea reiniciar la aplicación a sus valores por defecto eliminando los datos almacenados o no hacerlo.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.2.2. ReiniciarControlador.ListenerReiniciarRueda

/**

* Interfaz que permite comunicar al controlador si el usuario ha decidido reiniciar los valores de la aplicación o no.

*/

- **void confirmarReinicio():**

/**

* Función que se encargará de confirmar el reinicio de los valores de la aplicación.

*/

- **void cancelarReinicio():**

/**

* Función que se encargará de cancerlar el reinicio de los valores de la aplicación.

*/

2.2.2.3. ReiniciarModelo

/**

* Modelo del modal correspondiente a reiniciar la aplicación.

* Muestra un modal que permitirá al usuario decidir si desea reiniciar la aplicación a sus valores por defecto eliminando los datos almacenados o no hacerlo.

*

* @author Marcos Unzueta Puente @srunzu15 en github

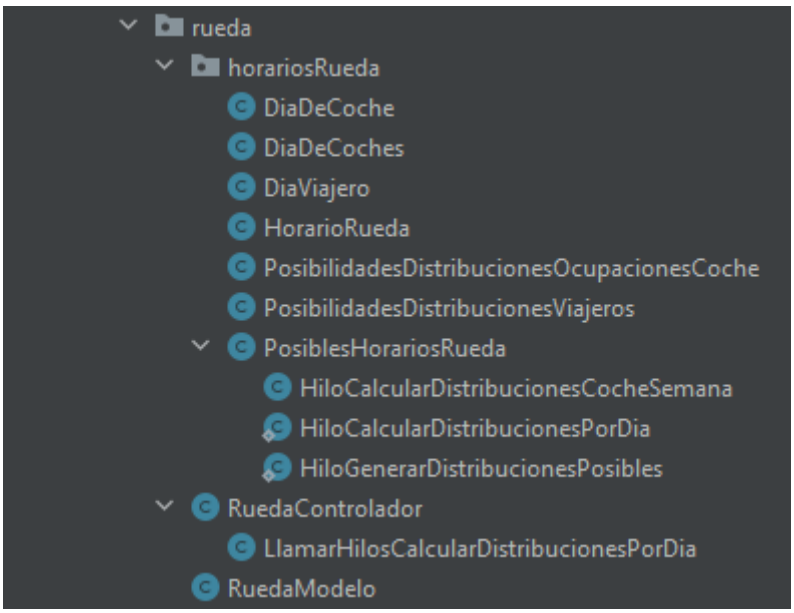
* @version 1.0

*/

2.2.3. Rueda

En la carpeta Rueda se encuentran los ficheros que se encargan de calcular el horario de rueda óptimo y de generar el PDF.

Tabla 7: Esquema de java.com.marcosunzueta.larueda.rueda



2.2.3.1. horariosRueda/DiaDeCoche:

/**

* Día del coche que se desplaza al centro un día concreto.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **int conductor:** Conductor del vehículo.
- **List<DiaViajero> viajerosEntrada:** Viajeros que se desplazan en el coche en la entrada.
- **List<DiaViajero> viajerosSalida:** Viajeros que se desplazan en el coche en la salida.
- **HoraRueda horaEntrada:** Hora a la que entra el coche.
- **HoraRueda horaSalida:** Hora a la que sale el coche.
- **int minutosEsperados:** Minutos esperados por los viajeros que se desplazan en el coche.

- **public DiaDeCoche(Integer conductor, List<Integer> viajerosEntradaIDs):**

  /**

  * Constructor

  *

  * @param conductor ID del conductor del vehículp.

  * @param viajerosEntradaIDs IDs de los viajeros que se desplazan en la entrada en el vehículo.

  */

- **public DiaDeCoche(DiaDeCoche coche, List<Integer> viajerosVuelta):**

  /**

  * Constructor

  *

  * @param coche Dia de coche del que se extraerán los datps de la ida.

  * @param viajerosVuelta IDs de los viajeros que se desplazan en la vuelta del vehículo.

  */

- **public DiaDeCoche(DiaDeCoche diaDeCoche):**

  /**

  * Constructor

  *

  * @param diaDeCoche Dia de coche del que se extraerá la información.

  */

- **public void establecerHoraEntrada(List<DiaViajero> viajeros):**

  /**

  * Teniendo en cuenta los viajeros que se desplazan en la ida del coche, calculará la hora de entrada para que todos lleguen al trabajo puntuales.

  * La hora de entrada se corresponderá con aquella que sea más temprano de todos los viajeros.

  *

  * @param viajeros Viajeros que se desplazarán ese día.

  */

- **void establecerHoraEntrada(List<DiaViajero> viajerosSinAsignar, List<Integer> idsViajerosSinViajero):**

  /**

  * Teniendo en cuenta los viajeros que se desplazan en la ida del coche, calculará la hora de entrada para que todos lleguen al trabajo puntuales.

* Es la parte iterativa de la función.

*

* @param viajerosSinAsignar Viajeros que aún no se han asignado.

* @param idsViajerosSinViajero IDs de los que aún no se ha encontrado viajero.

*/

- **void establecerHoraSalida(List<DiaViajero> viajeros):**

/**

* Teniendo en cuenta los viajeros que se desplazan en la vuelta del coche calculará la hora de salida de este.

*

* @param viajeros Viajeros que se desplazarán ese día.

*/

- **void establecerHoraSalida(List<DiaViajero> viajerosSinAsignar, List<Integer> idsViajerosSinViajero):**

/**

* Teniendo en cuenta los viajeros que se desplazan en la vuelta del coche, calculará la hora de salida .

* Es la parte iterativa de la función.

*

* @param viajerosSinAsignar Viajeros que aún no se han asignado.

* @param idsViajerosSinViajero IDs de los que aún no se ha encontrado viajero.

*/

- **void calcularMinutosEsperadosEntrada():**

/**

* Se calculan los minutos esperados por los viajeros de un coche en la entrada.

*/

- **void calcularMinutosEsperadosSalida():**

/**

* Se calculan los minutos esperados por los viajeros de un coche en la salida.

*/

2.2.3.2. horariosRueda/DiaDeCoches:

/**

* Se corresponde con un único día de rueda.

* De esta manera, almacena todos los coches que se desplazan ese día.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **coches:** Viajeros que se desplazan en el coche en la entrada.
- **minutosEsperados**: Minutos que esperan los viajeros fuera de su horario.
- DiaDeCoches(List<DiaDeCoche> coches):

  /**

  * Constructor que genera una lista de coches instanciando nuevos objetos con los datos dados.

  *

  * @param coches Coches que se desplazan.

  */
- **void calcularMinutosEsperadosEntrada():**

  /**

  * Se calculan los minutos esperados por los viajeros de cada coche en la entrada.

  */
- **void calcularMinutosEsperadosSalida():**

  /**

  * Se calculan los minutos esperados por los viajeros de cada coche en la salida.

  */
- **void calcularMinutosEsperados():**

  /**

  * Se calculan los minutos esperados por los viajeros de cada coche.

  */
- **void ordenarCochesPorHoraDeEntrada():**

  /**

  * Ordena la lista de coches en función de su hora de entrada.

  */
- **void ordenarCochesPorHoraDeSalida():**

  /**

  * Ordena la lista de coches en función de su hora de salida.

  */

2.2.3.3. horariosRueda/DiaViajero

/**

* Horario del viajero un día concreto.

* Esta clase se usará en los algoritmos de cálculo de rueda, de tal manera que cuando se esté calculando el mejor horario de un día concreto no haga falta consultar toda la información del usuario.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **int idViajero:** Id del viajero al que corresponde el día.
- **int dia:** Día correspondiente.
- **EntradaSalida entradaSalidaDelCentro:** Horario de entrada y salida del centro.
- **DiaViajero(Viajero viajero, int dia):**

  /**

  * Constructor del horario del viajero para un día concreto.

  * @param viajero Viajero del que se obtendrán los daros.

  * @param dia Día de la semana al que se corresponde el DiaViajero.

  */

2.2.3.4. horariosRueda/HorarioRueda:

/**

* Almacena la información correspondiente a un horario semanal de rueda.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **List<DiaDeCoches> diasDeCoches:** Se almacenan los siete días de coches de la semana.
- **int minutosEsperados:** Minutos esperados por los viajeros a lo largo de la semana.
- **int cochesTotalesLlevados:** Coches que se han llevado a lo largo de la semana.
- **int bondad:** Bondad del horario.
- **final Map<Integer, Integer> cochesLlevados:** Mapa que contiene los coches llevados por cada viajero.

- **HorarioRueda(List<Viajero> viajeros):**

  /**

  * Constructor.

  *

  * @param viajeros Viajeros que se desplazarán en la rueda a lo largo de la semana.

  */

- **HorarioRueda(HorarioRueda horarioRueda):**

  /**

  * Constructor.

  *

  * @param horarioRueda al partir del cual se generará otro objeto.

  */

- **boolean anadirDiaDeCoches(DiaDeCoches diaDeCoches):**

  /**

  * Añade un dia de coches al horario y comprueba si el horario sigue siendo viable.

  *

  * @param diaDeCoches Dia de coches a añadir.

  * @return Indica si el horario generado es viable.

  */

- **void calcularBondad():**

  /**

  * Se calcula la bondad del horario.

  */

2.2.3.5. horariosRueda/PosibilidadesDistribucionesOcupacionesCoche:

/**

* Almacenará todas las distribuciones en coches posibles teniendo en cuenta el número de viajeros.

*

* Esta clase se ocupará de calcular las posibilidades de distribución en número de coches y número de viajeros por coche, pero aún no tendrá en cuenta quién viaja en cada coche.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **List<DiaViajero> viajeros:** Viajeros que se desplazan ese día y sus horarios.
- **PosibilidadesDistribucionesViajeros distribucionesDeViajerosPosibles:** Lista de posibles distribuciones.
- **List<DiaDeCoches> diasDeCoches:** Días de coches.
- **int numeroMinimoCoches:** Número mínimo de coches que se desplazarán ese día.
- **int numeroMaximoCoches:** Número máximo de coches que se desplazarán ese día.
- **PosibilidadesDistribucionesOcupacionesCoche():**

  /**

  * Constructor de la clase.

  * Instancia la lista de viajeros, de tal manera que posteriormente se puedan añadir viajeros a la lista mediante el método add.

  */
- **void calcularMaximoYMinimoNumeroDeCoches():**

  /**

  * Calcula el máximo y el mínimo número de coches que se podrán desplazar en un día.

  */
- **void generarDistribucionesPosibles():**

  /**

  * Genera las posibles distribuciones en coches para un número de viajeros, valorando todas las posibilidades de número de coches desplazados y número de viajeros en cada coche.

  *

  */
- **void calcularTodasLasPosiblesDistribucionesDelNumeroDeViajerosEnCoches(List<Integer> posibilidadesOcupacionCoche, int numeroTotalViajeros, int numeroDeCoches):**

  /**

  * Calcula todas las posibles de distribución del número de viajeros en coches.

  *

  * @param posibilidadesOcupacionCoche Lista con los posibles números de viajeros por coche.

  * @param numeroTotalViajeros Número de viajeros a distribuir.

  * @param numeroDeCoches Número de coches en los que se tendrán que distribuir los viajeros.

*/

- **void calcularTodasLasPosiblesDistribucionesDelNumeroDeViajerosEnCoches(List<Integer> posibilidadesOcupacionCoche, int numeroTotalViajeros, List<Integer> listaParcial, int numeroDeCoches):**

  /**

  * Función iterativa mediante la cual se calcularán todas las posibles de distribución del número de viajeros en coches.

  *

  * @param posibilidadesOcupacionCoche Lista con los posibles números de viajeros por coche.

  * @param numeroTotalViajeros Número de viajeros a distribuir.

  * @param listaParcial Lista parcial con los viajeros que ya se encuentran distribuidos en coches. Esta lista se seguirá completando.

  * @param numeroDeCoches Número de coches en los que se tendrán que distribuir los viajeros.

  */

- **void calcularLasDistribucionesPosiblesDeLosViajerosEnLosDistintosCoches():**

  /**

  * Una vez se ha calculado de manera numérica cuántas personas han de viajar en cada coche se procede a calcular quienes serán esas personas.

  */

2.2.3.6. horariosRueda/PosibilidadesDistribucionesViajeros

/**

* Almacenará todas las distribuciones posibles de los viajeros en un número de coches dado con una cantidad de viajeros por coche determinada.

*

* Esta clase se ocupará de calcular las posibilidades de distribución en cuanto a qué viajeros concretos viajarán en cada coche.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **List<DiaViajero> viajeros:** Viajeros que se desplazan ese día y sus horarios.

- **List<Integer> idsViajeros:** IDs de los viajeros que se desplazan ese día.

- **List<List<Integer>> distribucionesEnCoches:** Distribución de los viajeros en coches de manera numérica.

- **List<List<List<Integer>>> posiblesDistribucionesPorCocheYNumeroDeViajeros:** Todas las distribuciones de viajeros posibles para todos los tamaños de coche posibles (Todas las posibles ocupaciones para un coche de dos personas, para un coche de tres ...).

- **List<List<List<Integer>>> posiblesDiasDeCocheSinConductor:** Todos los diasDeCoche viables.

- **List<DiaDeCoches> diasDeCoches:** Contiene todos los posibles días de coches, será el resultado.

- **PosibilidadesDistribucionesViajeros(List<DiaViajero> viajeros):**

  /**

  * Constructor que se encargará de instanciar la clase.

  *

  * @param viajeros Viajeros que se desplazarán ese día.

  */

- **List<DiaDeCoches> calcularTodasLasPosiblesDistribucionesDelNumeroDeViajerosEnCoches():**

  /**

  * Se procede a calcular quienes serán las personas que viajen en cada coche.

  * Para llamar a esta función es necesario que se haya rellenado previamente la lista de distribucionesEnCoches.

  */

- **List<List<Integer>> calcularTodasLasPosibilidadesParaUnaOcupacionDeCoche(int posibleOcupacion):**

  /**

  * Función que se encargará de calcular todas las posibilidades existentes para ocupar un coche que llevará un número de viajeros determinado.

  *

  * @param posibleOcupacion Número de viajeros

  * @return Todas las posibles combinaciones de ocupaciones del coche para ese número de viajeros.

  */

  - **void calcularTodasLasPosibilidadesParaUnaOcupacionDeCoche(int posibleOcupacion, List<Integer> diaDeCocheParcial, List<List<Integer>> listaDePosibilidades, List<Integer> viajerosPorValorar):**

  /**

* Función iterativa que se ayudará a calcular todas las posibilidades existentes para ocupar un coche que llevará un número de viajeros determinado.

*

* @param posibleOcupacion Número de viajeros.

* @param diaDeCocheParcial Estado actual del coche sobre el que se iterará.

* @param listaDePosibilidades Lista con las posibilidades de ocupar el coche. Conformará el resultado.

* @param viajerosPorValorar Viajeros que se pueden considerar para ocupar el coche.

*/

- **List<List<List<Integer>>> calcularTodasLasDistribucionesPosiblesDeLosViajerosEnLosCoches(List<Integer> diaCoches):**

/**

* Dado un dia de coches en ocupaciones (p.ej [3,2,1]) devuelve todas las posibilidades de distribución de los distintos viajeros en estos.

*

* @param diaCoches Dia de coches en número de viajeros (p.ej [3,2,1] serán tres coches uno con 3 viajeros, otro con 2 y otro con 1).

* @return Devuelve una lista con todas las combinaciones de viajeros posible para los tamaños dados. (p.ej {[(1,2,3),(4,5),(6)], [(1,2,3),(4,6),(5)]...})

*/

- **void calcularTodasLasDistribucionesPosiblesDeLosViajerosEnLosCoches(List<List<Integer> > diaCochesActual, List<List<List<Integer>>> diasDeCoches, List<Integer> cochesPendientes, List<List<List<Integer>>> posiblesDistribucionesPorCocheYNumeroDeViajerosDisponibles):**

/**

* Función iterativa que ayuda a que dado un dia de coches en ocupaciones se devuelvan todas las posibilidades de distribución de los distintos viajeros en estos.

*

* @param diaCochesActual Lista con los dias de coche con viajeros asignados que generará el resultado.

* @param diasDeCoches Lista con los dias de coches viables.

* @param cochesPendientes Lista con los coches que quedan por rellenar.

*/

- **void asignarConductorATodosLosDiasDeCochesPosibles(List<List<Integer>> diaDeCoches):**

/**

* Habiendo calculado todas las posibles distribuciones de los días de coches hay que asignar los conductores de cada vehículo.

*/

- **void asignarConductorATodosLosDiasDeCochesPosibles(List<DiaDeCoche> diaDeCochesActual, List<List<Integer>> cochesDelDiaSinConductor):**

  /**

  * Habiendo calculado todas las posibles distribuciones de los días de coches hay que asignar los conductores de cada vehículo.

  * Esta es la función iterativa.

  *

  * @param diaDeCochesActual Dia de coches que se está generando de manera iterativa. Los coches que estén en esta lista ya tendrán asignados conductores.

  * @param cochesDelDiaSinConductor Coches a los que aún no se les ha asignado conductor.

  */

- **void calcularTodasLasPosiblesVueltasParaUnaIdaYSeleccionarLaMejor(DiaDeCoches diaDeCoches):**

  /**

  * Se encargará de generar todas las vueltas posibles para una ida determinada y seleccionará la mejor.

  *

  * @param diaDeCoches Ida del dia de coches.

  */

  - **List<List<List<Integer>>> calcularTodasLasDistribucionesPosiblesDeLosViajerosEnLosCochesVuelta(List<Integer> diaCoches, List<List<List<Integer>>> posiblesDistribucionesPorCocheYNumeroDeViajerosDadosLosConductores):**

    /**

    * Dado un dia de coches en ocupaciones (p.ej [3,2,1]) devuelve todas las posibilidades de distribución de los distintos viajeros en estos.

    *

    * @param diaCoches Dia de coches en número de viajeros (p.ej [3,2,1] serán tres coches uno con 3 viajeros, otro con 2 y otro con 1).

    * @return Devuelve una lista con todas las combinaciones de viajeros posible para los tamaños dados.

    */

- **void calcularTodasLasDistribucionesPosiblesDeLosViajerosEnLosCochesVuelta(List<List<Integer>> diaCochesActual, List<List<List<Integer>>> diasDeCoches, List<Integer>**

cochesPendientes, List<List<List<Integer>>>
posiblesDistribucionesPorCocheYNumeroDeViajerosDisponibles):

/**

* Función iterativa que ayuda a que dado un día de coches en ocupaciones se devuelvan todas las posibilidades de distribución de los distintos viajeros en estos.

*

* @param diaCochesActual Lista con los días de coche con viajeros asignados sobre la que se iterará.

* @param diasDeCoches Lista con los días de coches viables.

* @param cochesPendientes Lista con los coches que quedan por rellenar.

*/

2.2.3.7. horariosRueda/PosiblesHorariosRueda

/**

* Almacena y genera todos los posibles horarios de rueda.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **List<HorarioRueda> horariosRueda:** Horarios de rueda generados.
- **List<Viajero> viajeros:** Lista de viajeros que se desplazarán a lo largo de la semana.
- **List<List<DiaDeCoches>> distribucionesOcupacionesCochesSemanaGenerado**
- **PosibilidadesDistribucionesOcupacionesCoche[] distribucionesOcupacionesCochesSemana:** Objetos encargados de calcular los posibles horarios correspondientes a cada día de la semana.
- **PosiblesHorariosRueda(List<Viajero> viajeros):**

  /**

  * Constructor que obtiene la lista de viajeros a partir de la cual se generará el horario de rueda óptimo.

  *

  * @param viajeros Viajeros a partir de los cuales se generará el horario.

  */

- **void distribuirViajerosEnLosDias():**

  /**

  * En cada instancia del día de la semana correspondiente se sitúan los viajeros que trabajan y su horario correspondiente a ese día.

*/

- **void calcularNumeroMaximoYMinimoDeCochesPorDia():**

  /**

  * Calcula el número máximo y el número mínimo de coches necesarios para realizar el trayecto correspondiente al día.

  */

- **List<HorarioRueda> generarPosiblesHorariosDeRuedaSecuencial():**

  Esta función no se usa actualmente, pero está en el proyecto para poder hacer pruebas.

  /**

  * Genera todos los horarios de rueda posibles de manera secuencial.

  *

  * @return todos los horarios de rueda posibles.

  */

- **void generarPosiblesHorariosDeRueda(HorarioRueda horarioRueda, List<List<DiaDeCoches>> distribucionesOcupacionesCochesSemana):**

  /**

  * Función iterativa que combinará todos los días de coches existente de cada día de la semana para generar todos los horarios viables.

  *

  * @param horarioRueda Horario de rueda que se está generando.

  * @param distribucionesOcupacionesCochesSemana Posibilidades valorables.

  */

- **List<HorarioRueda> generarPosiblesHorariosDeRuedaParalelo():**

  /**

  * Genera todos los horarios de rueda posibles usando un método paralelo.

  *

  * @return todos los horarios de rueda posibles.

  */

- **boolean LlamarHilosCalcularDistribucionesPorDia():**

  /**

  * Se calcularán las distribuciones posibles de los viajeros en coches cada día de la semana de manera paralela.

  *

* @return Se devolverá verdadero si la ejecución de todos los hilos se realiza de manera exitosa.

*/

- **boolean LlamarHilosCalcularDistribucionesCocheSemana():**

/**

* Distribuirá en hilos el trabajo de combinar todos los días de coches posibles calculados de cada día de la semana para calcular todos los horarios posibles.

* @return Devuelve verdadero o falso en función de si se ha podido hacer correctamente la distribución o no.

*/

2.2.3.8. horariosRueda/PosiblesHorariosRueda.HiloCalcularDistribucionesCocheSemana

/**

* Hilo que calcula los horarios posibles de las distribuciones que le corresponden.

*/

2.2.3.9. horariosRueda/PosiblesHorariosRueda.HiloCalcularDistribucionesPorDia

/**

* Hilo que calcula las distribuciones en coches posibles asignando IDS a cada distribución generada previamente .

*/

2.2.3.10. horariosRueda/PosiblesHorariosRueda.HiloCalcularDistribucionesPosibles

/**

* Hilo que calcula las distribuciones en coches posibles (en número de viajeros) .

*/

2.2.3.11. RuedaControlador

/**

* Controlador de la actividad correspondiente a la sección de rueda.

* Se encarga de calcular el horario de rueda óptimo, así como de mostrarlo.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **private void mostrarPDF():**

/**

* Comprueba si existe algún PDF de la rueda y modifica el aspecto de la actividad en consecuencia.

*/

2.2.3.12. RuedaControlador.LlamarHilosCalcularDistribucionesPorDia

/**

* Función asíncrona que calculará el horario de rueda óptimo sin sobrecargar el hilo principal.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.3.13. RuedaModelo

/**

* Modelo de la actividad correspondiente a la sección de rueda.

* Se encarga de calcular el horario de rueda óptimo, así como de mostrarlo.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **List<HorarioRueda> ruedasGeneradas:** Ruedas generadas mediante el algoritmo.
- **List<Viajero> viajeros:** Lista de viajeros que formarán la rueda.
- **File ficheroPDF:** Fichero PDF de la rueda.
- **boolean hayViajerosRegistrados():**

  /**

  * Comprueba si hay viajeros registrados o no.

  *

  * @return Verdadero si hay viajeros registrados y falso si no.

  */
- **boolean recuperarHorarios():**

  /**

  * Recupera los horarios correspondientes a los viajeros.

  *

  * @return Verdadero si ha podido recuperar los horarios y falso si no.

  */
- **transicionAViajeros()**

/**

* Función que realiza una transición a la actividad de viajeros.

*/

- **boolean generarHorarioDeRueda()**

  /**

  * Genera el horario de rueda óptimo.

  *

  * @return Devuelve verdadero si ha podido generar el horario y falso si no.

  */

- **void generarPDFs()**

  /**

  * Genera los PDFs correspondientes a los horarios generados.

  */

- **float establecerTamanoTexto(Paint pincel, float anchoDisponible, String texto)**

  /**

  * Indica el tamaño de la letra de texto adecuada teniendo en cuenta el texto a escribir y el tamaño disponible.

  *

  * @param pincel Pincel en el que se establecerá el tamaño de letra.

  * @param anchoDisponible Ancho en el que hay que escribir el texto.

  * @param texto Texto a escribir.

  * @return Tamaño para un texto y un tamaño determinado.

  */

- **compartirPDF()**

  /**

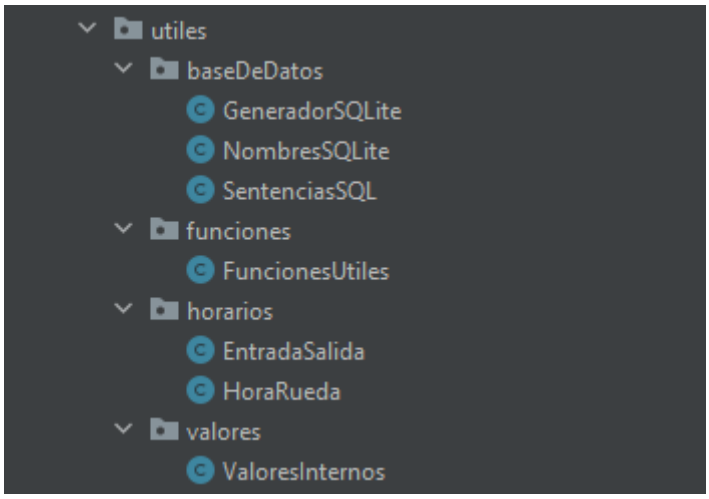
  * Comparte el PDF de la rueda generado.

  */

2.2.4. Utiles

Carpeta en la que se encuentran una serie de ficheros que pueden ser útiles para el desarrollo de la aplicación.

Tabla 8: : Esquema de java.com.marcosunzueta.larueda.utiles



2.2.4.1. basesDeDatos/GeneradorSQLite

/**

* Clase que generará la base de datos local que usará la aplicación.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.4.2. basesDeDatos/NombreSQLite

/**

* Clase que almacenará los nombres de las tablas y atributos de la base de datos.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.4.3. basesDeDatos/SentenciasSQL

/**

* Clase que almacenará las sentencias SQL que usará la aplicación.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.4.4. funciones/FuncionesUtiles

/**

* Clase que almacenará funciones que serán accesibles a todas las clases de la aplicación.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

SQLiteDatabase obtenerReferenciaBDEscritura(Context contexto):

/**

* Genera la referencia a la base de datos en modo escritura.

*

* @param contexto Contexto de la aplicación.

* @return Referencia a la base de datos en modo escritura.

*/

- **String crearSentenciaNuevoViajero(int idViajero, String nombre, String primerApellido, String segundoApellido):**

  /**

  * Crea la sentencia SQL que permite añadir un nuevo viajero a la base de datos.

  *

  * @param idViajero ID del usuario a crear.

  * @param nombre Nombre del viajero. (No puede ser null)

  * @param primerApellido Primer apellido del viajero.

  * @param segundoApellido Segundo apellido del viajero.

  * @return Sentencia SQL que añadirá el viajero deseado a la base de datos.

  */

- **String crearSentenciaNuevoDiaViajeroLaboral(int idViajero, String dia, Integer horaEntrada, Integer minutoEntrada, Integer horaSalida, Integer minutoSalida):**

  /**

  * Crea la sentencia SQL que permite añadir un nuevo dia viajero a la base de datos.

  * El dia viajero corresponderá a un día laboral.

  *

  * @param idViajero ID del usuario al cual corresponde el día a crear.

  * @param dia Día correspondiente al horario. Ha de ser uno de los siguientes valores : "lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo".

  * @param horaEntrada Hora de entrada del viajero al trabajo.

  * @param minutoEntrada Minuto de entrada del viajero al trabajo.

* @param horaSalida Hora de salida del viajero del trabajo.

* @param minutoSalida Minuto de salida del viajero del trabajo.

* @return Sentencia SQL que añadirá el día viajero deseado a la base de datos.

*/

- **String crearSentenciaNuevoDiaViajeroNoLaboral(int idViajero, String dia):**

  /**

  * Crea la sentencia SQL que permite añadir un nuevo dia viajero a la base de datos.

  * El dia viajero corresponderá a un día no laboral.

  *

  * @param idViajero ID del usuario al cual corresponde el día a crear.

  * @param dia Día correpsondiente al horario. Ha de ser uno de los siguientes valores : "lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo".

  * @return Sentencia SQL que añadirá el día viajero deseado a la base de datos.

  */

- **String crearSentenciaEliminarViajero(int idViajero):**

  /**

  * Crea la sentencia SQL que permite eliminar un viajero de la base de datos.

  *

  * @param idViajero ID del viajero a eliminar.

  * @return Sentencia SQL que eliminará el viajero deseado de la base de datos.

  */

- **String crearSentenciaRecuperarViajero(int idViajero):**

  /**

  * Crea la sentencia SQL que permite recuperar los datos del viajero deseado.

  *

  * @param idViajero ID del viajero del cual se recuperarán los datos.

  * @return Sentencia SQL que recuperará los datos del viajero deseado de la base de datos.

  */

- **String crearSentenciaRecuperarDiasViajeroDeViajero(int idViajero):**

  /**

  * Crea la sentencia SQL que permite recuperar los días viajero correspondientes a un viajero de la base de datos.

  *

* @param idViajero ID del viajero del cual se recuperarán los días.

* @return Sentencia SQL que recuperará los días viajero correspondientes de la base de datos.

*/

- **String crearSentenciaModificarViajero(int idViajero, String nombre, String primerApellido, String segundoApellido):**

/**

* Crea la sentencia SQL que permite modificar los datos correspondientes al viajero de la base de datos.

*

* @param idViajero ID del viajero del cual se modificarán los datos.

* @param nombre Nuevo nombre del viajero.

* @param primerApellido Nuevo primer apellido del viajero.

* @param segundoApellido Nuevo segundo apellido del viajero.

* @return Sentencia SQL que modificará los datos del viajero correspondientes de la base de datos.

*/

- **String crearSentenciaModificarDiaViajeroLaboral(int idViajero, String dia, Integer horaEntrada, Integer minutoEntrada, Integer horaSalida, Integer minutoSalida):**

/**

* Crea la sentencia SQL que permite modificar los datos correspondientes a un día viajero de la base de datos.

* El dia viajero a modificar se corresponderá con un día laboral.

*

* @param idViajero ID del viajero del cual se modificarán los datos.

* @param dia Día correspondiente al horario del cual se modificarán los datos. Ha de ser uno de los siguientes valores : "lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo".

* @param horaEntrada Nueva hora de entrada del viajero al trabajo.

* @param minutoEntrada Nuevo minuto de entrada del viajero al trabajo.

* @param horaSalida Nueva hora de salida del viajero del trabajo.

* @param minutoSalida Nuevo minuto de salida del viajero del trabajo.

* @return Sentencia SQL que modificará los datos de un dia viajero correspondiente en la base de datos.

*/

- **String crearSentenciaModificarDiaViajeroNoLaboral(int idViajero, String dia):**

/**

* Crea la sentencia SQL que permite modificar los datos correspondientes a un día viajero de la base de datos.

* El día viajero a modificar se corresponderá con un día no laboral.

*

* @param idViajero ID del viajero del cual se modificarán los datos.

* @param dia Día correspondiente al horario del cual se modificarán los datos. Ha de ser uno de los siguientes valores : "lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo".

* @return Sentencia SQL que modificará los datos de un día viajero correspondiente en la base de datos.

*/

- **SharedPreferences recuperarAlmacenamientoInternoIds(Activity actividad):**

  /**

  * Recupera el SharedPreferences que contiene los ids que se autoincrementarán.

  *

  * @param actividad Actividad que pretende recuperar el SharePreferences.

  * @return SharedPreferences que contiene los ids que se autoincrementarán.

  */

- **boolean horaSalidaMayorQueDeEntrada(int horaEntrada, int minutoEntrada, int horaSalida, int minutoSalida):**

  /**

  * A partir de unos valores de entrada y unos de salida calcula si la salida es posterior a la entrada o no.

  *

  * @param horaEntrada Hora de entrada del viajero al trabajo.

  * @param minutoEntrada Minuto de entrada del viajero al trabajo.

  * @param horaSalida Hora de salida del viajero al trabajo.

  * @param minutoSalida Minuto de salida del viajero del trabajo.

  * @return Devuelve verdadero si la hora de salida es posterior a la de entrada y falso si no.

  */

- **String crearSentenciaModalEliminarViajero(Activity controlador, int idUsuario, String nombre):**

  /**

* Crea el texto que se mostrará en el modal de eliminación y que se usará para consultar al usuario si desea eliminar al viajero o no.

*

* @param controlador Controlador que mostrará el modal.

* @param idUsuario ID del usuario que se pretende eliminar.

* @param nombre Nombre del usuario que se pretende eliminar.

* @return texto que se mostrará en el modal de eliminación y que se usará para consultar al usuario si desea eliminar al viajero o no.

*/

- **HorarioViajero recuperarHorarioDeViajero(Cursor cursor):**

  /**

  * Función que obtendrá el HorarioDeViajero correspondiente a partir de una consulta realizada en la base de datos.

  *

  * @param cursor Consulta a la base de datos.

  * @return Horario del viajero consultado.

  */

- **boolean comprobarHorarioNoEsNulo(Cursor cursor):**

  /**

  * Comprueba si el horario recuperado de la base de datos y correspondiente a un idUsuario/dia es nulo o no.

  *

  * @param cursor Consulta a la base de datos.

  * @return Devuelve verdadero si el horario no es nulo y falso si el horario es nulo.

  */

- **String traducirNumeroAValorDeEntrada(int numero):**

  /**

  * Traduce un número pasado al formato de texto utilizado para representar los horarios en los spinners de los formularios de viajeros.

  *

  * @param numero Número a traducir.

  * @return Numero traducido al formato de texto utilizado para representar los horarios en los spinners de los formularios de viajeros.

  */

- **List<Integer> generarListaDeNumeros(int inicio, int fin):**

/**

* Genera una lista de números enteros ordenados dándole el número de inicio y el de fin.

*

* @param inicio Primer número de la lista.

* @param fin Último número de la lista.

* @return Lista de números ordenados.

*/

- **List<Integer> interseccion(List<Integer> primeraLista, List<Integer> segundaLista):**

  /**

  * Calculcula la intersección entre dos listas.

  *

  * @param primeraLista Lista sobre la que se va a calcular la intersección.

  * @param segundaLista Lista sobre la que se va a calcular la intersección.

  * @return Intersección entre ambas listas.

  */

- **int comparaHoraRueda(HoraRueda hora1, HoraRueda hora2):**

  /**

  * @param hora1 Hora de rueda a comparar.

  * @param hora2 Hora de rueda a comparar.

  * @return Diferencia del horario de rueda 1 respecto al de 2 en minutos.

  */

- **String formatearNumeroParaQueTengaDOsDigitos(Integer numero):**

  /**

  * Formatea un número para que se muestre siempre como un número de dos dígitos (00, 01 ...)

  *

  * @param numero Número a formatear

  * @return Número formateado.

  */

- **Viajero conocerViajeroAPartirdelID(List<Viajero> viajeros, int idViajero):**

  /**

  * Indica el viajero al que se refiere un id concreto.

  *

* @param viajeros Viajeros entre los que se buscará.

* @param idViajero Id del viajero a buscar,

* @return Viajero buscado.

*/

2.2.4.5. horarios/EntradaSalida

/**

* Modelo que almacenará la información de las horas de entrada y salida correspondientes a un día.

* Esta información es aplicable para un viajero, para un coche, etc.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **HoraRueda entrada:** Horario de entrada del día.
- **HoraRueda salida:** Horario de salida del día.
- **boolean laboral:** Indica si el día es laboral o no.
- **EntradaSalida(HoraRueda entrada, HoraRueda salida, boolean laboral):**

/**

* Constructor del objeto que almacenará la información de las horas de entrada y de salida correspondientes a un día.

* @param entrada Hora y minuto de entrada.

* @param salida Hora y minuto de salida.

* @param laboral Indica si el día es laboral o no.

*/

2.2.4.6. horarios/HoraRueda

/**

* Modelo que almacenará la información referente a una entrada/salida.

* En la aplicación solo nos interesa la hora y el minuto en el que se entra o sale del centro de trabajo.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **int hora**

- **int minuto**

- **HoraRueda(int hora, int minuto):**
  /**
  * Constructor de la hora de rueda, la cual tan solo necesita la hora y el minuto concreto.
  *
  * @param hora Hora.
  * @param minuto Minuto.
  */

- **HoraRueda(HoraRueda horaRueda):**

  /**

  * Constructor de la hora de rueda, generará una nueva instancia de las clases usando los mismos valores que el objeto pasado como parámetro.

  *

  * @param horaRueda Objeto a partir del cual se quieren extraer los datos.

  */

2.2.4.7. valores/ValoresInternos

/**

* Clase que almacenará los valores internos de la aplicación.

* Se consideran valores internos a todos aquellos que son usados por las distintas clases de la aplicación y no se modificarán en función del idioma del usuario.

*

* @author Marcos Unzueta Puente @srunzu15 en github

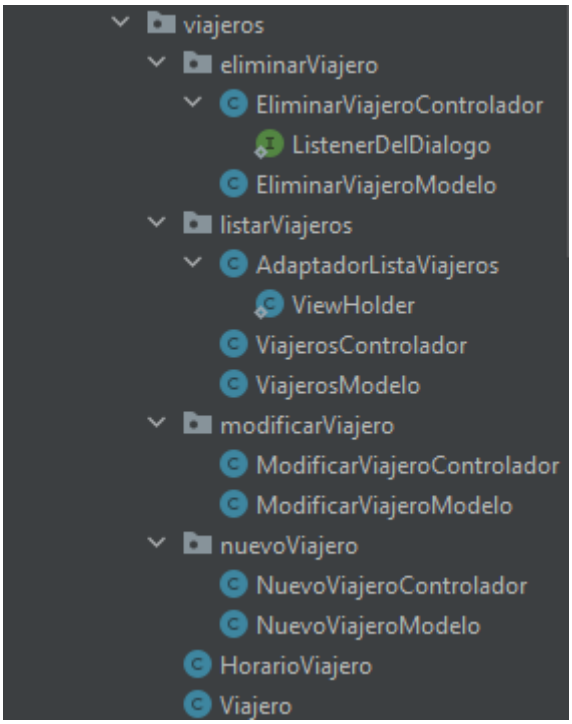
* @version 1.0

*/

2.2.5. Viajeros

En la carpeta Rueda se encuentran los ficheros que se encargan de la gestión de los datos de los viajeros.

Tabla 9: Esquema de java.com.marcosunzueta.larueda.viajeros



2.2.5.1. eliminarViajero/EliminarViajeroControlador:

/**

* Controlador del modal correspondiente a eliminar viajero.

* Muestra un modal que permitirá al sistema cerciorarse sobre si el usuario desea eliminar al viajero correspondiente.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.5.2. eliminarViajero/ EliminarViajeroControlador.ListenerDelDialogo:

/**

* Interfaz que permite comunicar al controlador si el usuario ha decidido eliminar al viajero o no.

*/

- **void confirmarEliminacion():**

/**

* Función que se encargará de confirmar la decisión de eliminar al viajero.

*/

- **void cancelarEliminacion():**

/**

* Función que se encargará de cancelar la decisión de eliminar al viajero.

*/

2.2.5.3. eliminarViajero/EliminarViajeroModelo:

/**

* Modalo del modal correspondiente a eliminar viajero.

* Muestra un modal que permitirá al sistema cerciorarse sobre si el usuario desea eliminar al viajero correspondiente.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **String nombre:** Nombre del viajero a eliminar.
- **int id:** ID del viajero a eliminar.
- **recuperarNombreEId(Bundle argumentos):**

/**

* Función que recuperará de los argumentos pasados al modal el nombre y el id del usuario que se pretende eliminar.

* @param argumentos Argumentos pasados al modal.

*/

- **String generarTextoEliminarViajero():**

/**

* Función que genera el texto que se mostrará como aviso al usuario para verificar la eliminación del viajero.

* @return texto a mostrar.

*/

2.2.5.4. listarViajeros/AdaptadorListaViajeros:

/**

* Aadaptador que se utilizará para dar formato al recycler que muestra a los viajeros almacenados en la aplicación.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **List<Viajero> listaDeViajeros:** Lista de viajeros que se mostrarán.

- **LayoutInflater inflater**

- **AdaptadorListaViajeros(List<Viajero> listaDeViajeros, Context contexto, ViajerosModelo modelo):**

  /**
  * Constructor del adaptador que se utilizará para dar formato al recycler que muestra a los viajeros almacenados en la aplicación.
  * @param listaDeViajeros Lista de viajeros que se mostrarán.
  * @param contexto Controlador que llama al modal.
  * @param modelo Modelo que contiene los datos necesarios para que el modal funcione correctamente.
  */

2.2.5.5. listarViajeros/AdaptadorListaViajeros.ViewHolder:

/**

* ViewHolder que se encargará de gestionar cada elemento de la lista.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

2.2.5.6. listarViajeros/ViajerosControlador

/**

* Controlador de la actividad viajeros.

* Muestra a todos los viajeros que forman parte de la rueda en una lista.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **void inicializarListaDeViajeros():**

  /**

  * Función que se encargará de dar el formato correspondiente a la vista, en función de si hay viajeros registrados en la base de datos o si no.

  */

- **void viajeroNoEliminado():**

  /**

  * Función que indicará al usuario que el viajero no se ha eliminado.

  */

- **void viajeroEliminado():**

/**

* Función que indicará al usuario que el viajero se ha eliminado.

* También vuelve a leer los viajeros de la base de datos y actúa en consecuencia.

*/

2.2.5.7. listarViajeros/ViajerosModelo:

/**

* Modelo de la actividad viajeros.

* Muestra a todos los viajeros que forman parte de la rueda en una lista.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **void transicionANuevoViajero():**

  /**

  * Función que realiza una transición a la actividad de nuevo viajero.

  */

- **void transicionAModificarViajero(int idViajero):**

  /**

  * Función que realiza una transición a la actividad de modificar viajero.

  *

  * @param idViajero ID del viajero a modificar.

  */

- **void consultarEliminarViajero(int idViajero, String nombreViajero):**

  /**

  * Función que mostrará al usuario un modal consultándole si desea eliminar al viajero que ha seleccionado previamente.

  *

  * @param idViajero ID del viajero a eliminar.

  * @param nombreViajero Nombre del viajero a eliminar.

  */

- **boolean inicializarRecyclerDeViajeros(RecyclerView recyclerView):**

  /**

  * Comprueba si hay viajeros almacenados en la base de datos, y en caso de que así sea, rellena el RecyclerView.

*

* @param recyclerView Recycler View en el que se muestran listados todos los viajeros.

* @return Devuelve verdadero o falso en función de si existen viajeros almacenados en la base de datos y si estos se han podido mostrar en el recycler o no.

*/

- **void eliminarViajeroSeleccionado():**

  /**

  * Función que se encarga de eliminar el viajero seleccionado.

  */

2.2.5.8. modificarViajero/ModificarViajeroControlador:

/**

* Controlador de la actividad modificar viajero.

* Muestra los datos del viajero seleccionado y permite modificarlos.

*

@author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **void marcarDiaComoLaboral(MaterialSpinner entradaHoras, MaterialSpinner entradaMinutos, MaterialSpinner salidaHoras, MaterialSpinner salidaMinutos):**

  /**

  * Permite maquetar los elementos correspondiente a un día concreto, de tal manera que, permita interpretar al usuario que es laboral.

  *

  * @param entradaHoras Spinner correspondiente a la hora de entrada.

  * @param entradaMinutos Spinner correspondiente a los minutos de entrada.

  * @param salidaHoras Spinner correspondiente a la hora de salida.

  * @param salidaMinutos Spinner correspondiente a los minutos de salida.

  */

- **void marcarDiaComoNoLaboral(MaterialSpinner entradaHoras, MaterialSpinner entradaMinutos, MaterialSpinner salidaHoras, MaterialSpinner salidaMinutos):**

  /**

  * Permite maquetar los elementos correspondiente a un día concreto, de tal manera que, permita interpretar al usuario que no es laboral.

  *

  * @param entradaHoras Spinner correspondiente a la hora de entrada.

* @param entradaMinutos Spinner correspondiente a los minutos de entrada.

* @param salidaHoras Spinner correspondiente a la hora de salida.

* @param salidaMinutos Spinner correspondiente a los minutos de salida.

*/

- **void viajeroModificado():**

/**

* Mensaje que se mostrará cuando se modifiquen los valores de un viajero en la base de datos.

*/

- **void viajeroNoModificado():**

/**

* Mensaje que se mostrará cuando no se modifiquen los valores de un viajero en la base de datos.

*/

- **void errorCamposObligatoriosVacios():**

/**

* Error que se muestra cuando se ha dejado sin rellenar un campo obligatorio.

*/

- **void errorViajeroNoTrabajaNingunDia():**

/**

* Error que se muestra cuando no se establece horario de trabajo ningún día del viajero.

*/

- **void errorHoraDeSalidaNoEsAnteriorAHoraDeEntrada():**

/**

* Error que se muestra cuando la hora de salida introducida es anterior a la hora de entrada.

*/

2.2.5.9. modificarViajero/ModificarViajeroModelo

/**

* Modelo de la actividad modificar viajero.

* Muestra los datos del viajero seleccionado y permite modificarlos.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **int idViajero:** ID del viajero a modificar.
- **Viajero viajero:** Viajero a modificar.
- **void recuperarIdViajero(Intent intent):**

  /**

  * Función que recuperará el id de usuario pasado como parámetro.

  *

  * @param intent Parámetros pasados desde la actividad anterior.

  */

- **boolean recuperarViajeroDeLaBaseDeDatos():**

  /**

  * Se recuperan los datos correspondientes al viajero de la base de datos.

  *

  * @return Verdadero si se han recuperado los datos del viajero correctamente y falso si no es así.

  */

- **boolean modificarViajero(String nombre, String primerApellido, String segundoApellido, boolean lunesLaboral, String lunesEntradaHoras, String lunesEntradaMinutos, String lunesSalidaHoras, String lunesSalidaMinutos, boolean martesLaboral, String martesEntradaHoras, String martesEntradaMinutos, String martesSalidaHoras, String martesSalidaMinutos, boolean miercolesLaboral, String miercolesEntradaHoras, String miercolesEntradaMinutos, String miercolesSalidaHoras, String miercolesSalidaMinutos, boolean juevesLaboral, String juevesEntradaHoras, String juevesEntradaMinutos, String juevesSalidaHoras, String juevesSalidaMinutos, boolean viernesLaboral, String viernesEntradaHoras, String viernesEntradaMinutos, String viernesSalidaHoras, String viernesSalidaMinutos, boolean sabadoLaboral, String sabadoEntradaHoras, String sabadoEntradaMinutos, String sabadoSalidaHoras, String sabadoSalidaMinutos, boolean domingoLaboral, String domingoEntradaHoras, String domingoEntradaMinutos, String domingoSalidaHoras, String domingoSalidaMinutos):**

  /**

  * Función que modificará los datos del viajero en la base de datos.

  *

  * @param nombre Nombre del viajero.

  * @param primerApellido Primer apellido del viajero.

  * @param segundoApellido Segundo apellido del viajero.

  * @param lunesLaboral Indica si el lunes es día laboral o no.

  * @param lunesEntradaHoras Hora de entrada del viajero el lunes al trabajo.

* @param lunesEntradaMinutos Minuto de entrada del viajero el lunes al trabajo.

* @param lunesSalidaHoras Hora de salida del viajero el lunes al trabajo.

* @param lunesSalidaMinutos Minuto de salida del viajero el lunes al trabajo.

* @param martesLaboral Indica si el martes es día laboral o no.

* @param martesEntradaHoras Hora de entrada del viajero el martes al trabajo.

* @param martesEntradaMinutos Minuto de entrada del viajero el martes al trabajo.

* @param martesSalidaHoras Hora de salida del viajero el martes al trabajo.

* @param martesSalidaMinutos Minuto de salida del viajero el martes al trabajo.

* @param miercolesLaboral Indica si el miércoles es día laboral o no.

* @param miercolesEntradaHoras Hora de entrada del viajero el miércoles al trabajo.

* @param miercolesEntradaMinutos Minuto de entrada del viajero el miércoles al trabajo.

* @param miercolesSalidaHoras Hora de salida del viajero el miércoles al trabajo.

* @param miercolesSalidaMinutos Minuto de salida del viajero el miércoles al trabajo.

* @param juevesLaboral Indica si el jueves es día laboral o no.

* @param juevesEntradaHoras Hora de entrada del viajero el jueves al trabajo.

* @param juevesEntradaMinutos Minuto de entrada del viajero el jueves al trabajo.

* @param juevesSalidaHoras Hora de salida del viajero el jueves al trabajo.

* @param juevesSalidaMinutos Minuto de salida del viajero el jueves al trabajo.

* @param viernesLaboral Indica si el viernes es día laboral o no.

* @param viernesEntradaHoras Hora de entrada del viajero el viernes al trabajo.

* @param viernesEntradaMinutos Minuto de entrada del viajero el viernes al trabajo.

* @param viernesSalidaHoras Hora de salida del viajero el viernes al trabajo.

* @param viernesSalidaMinutos Minuto de salida del viajero el viernes al trabajo.

* @param sabadoLaboral Indica si el sábado es día laboral o no.

* @param sabadoEntradaHoras Hora de entrada del viajero el sábado al trabajo.

* @param sabadoEntradaMinutos Minuto de entrada del viajero el sábado al trabajo.

* @param sabadoSalidaHoras Hora de salida del viajero el sábado al trabajo.

* @param sabadoSalidaMinutos Minuto de salida del viajero el sábado al trabajo.

* @param domingoLaboral Indica si el domingo es día laboral o no.

* @param domingoEntradaHoras Hora de entrada del viajero el domingo al trabajo.

* @param domingoEntradaMinutos Minuto de entrada del viajero el domingo al trabajo.

* @param domingoSalidaHoras Hora de salida del viajero el domingo al trabajo.

* @param domingoSalidaMinutos Minuto de salida del viajero el domingo al trabajo.

* @return Verdadero si el viajero se ha modificado correctamente y falso en caso contrario.

*/

2.2.5.10. nuevoViajero/NuevoViajeroControlador:

/**

* Controlador de la actividad que permitirá almacenar un viajero nuevo en la base de datos.

* Muestra un formulario con los datos a rellenar.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **void marcarDiaComoLaboral(MaterialSpinner entradaHoras, MaterialSpinner entradaMinutos, MaterialSpinner salidaHoras, MaterialSpinner salidaMinutos):**

  /**

  * Permite maquetar los elementos correspondiente a un día concreto, de tal manera que, permita interpretar al usuario que es laboral.

  *

  * @param entradaHoras Spinner correspondiente a la hora de entrada.

  * @param entradaMinutos Spinner correspondiente a los minutos de entrada.

  * @param salidaHoras Spinner correspondiente a la hora de salida.

  * @param salidaMinutos Spinner correspondiente a los minutos de salida.

  */

- **void marcarDiaComoNoLaboral(MaterialSpinner entradaHoras, MaterialSpinner entradaMinutos, MaterialSpinner salidaHoras, MaterialSpinner salidaMinutos):**

  /**

  * Permite maquetar los elementos correspondiente a un día concreto, de tal manera que, permita interpretar al usuario que no es laboral.

  *

  * @param entradaHoras Spinner correspondiente a la hora de entrada.

  * @param entradaMinutos Spinner correspondiente a los minutos de entrada.

  * @param salidaHoras Spinner correspondiente a la hora de salida.

  * @param salidaMinutos Spinner correspondiente a los minutos de salida.

  */

- **void viajeroRegistrado():**

  /**

  * Indica que el nuevo viajero se ha registrado en la base de datos.

  */

- **void viajeroNoRegistrado():**

  /**

  * Indica que el nuevo viajero no se ha registrado en la base de datos.

  */

- **void errorCamposObligatoriosVacios():**

  /**

  * Error que se muestra cuando se ha dejado sin rellenar un campo obligatorio.

  */

- **void errorViajeroNoTrabajaNingunDia():**

  /**

  * Error que se muestra cuando no se establece horario de trabajo ningún día del viajero.

  */

- **void errorHoraDeSalidaNoEsAnteriorAHoraDeEntrada():**

  /**

  * Error que se muestra cuando la hora de salida introducida es anterior a la hora de entrada.

  */

2.2.5.11. nuevoViajero/NuevoViajeroModelo:

/**

* Modelo de la actividad que permitirá almacenar un viajero nuevo en la base de datos.

* Muestra un formulario con los datos a rellenar.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **boolean almacenarViajero(String nombre, String primerApellido, String segundoApellido, boolean lunesLaboral, String lunesEntradaHoras, String lunesEntradaMinutos, String lunesSalidaHoras, String lunesSalidaMinutos, boolean martesLaboral, String martesEntradaHoras, String martesEntradaMinutos, String martesSalidaHoras, String martesSalidaMinutos, boolean miercolesLaboral, String miercolesEntradaHoras, String miercolesEntradaMinutos, String**

miercolesSalidaHoras, String miercolesSalidaMinutos, boolean juevesLaboral, String juevesEntradaHoras, String juevesEntradaMinutos, String juevesSalidaHoras, String juevesSalidaMinutos, boolean viernesLaboral, String viernesEntradaHoras, String viernesEntradaMinutos, String viernesSalidaHoras, String viernesSalidaMinutos, boolean sabadoLaboral, String sabadoEntradaHoras, String sabadoEntradaMinutos, String sabadoSalidaHoras, String sabadoSalidaMinutos, boolean domingoLaboral, String domingoEntradaHoras, String domingoEntradaMinutos, String domingoSalidaHoras, String domingoSalidaMinutos):

```
/**
* Función que añadirá al viajero en la base de datos.
*
* @param nombre                Nombre del viajero.
* @param primerApellido        Primer apellido del viajero.
* @param segundoApellido       Segundo apellido del viajero.
* @param lunesLaboral          Indica si el lunes es día laboral o no.
* @param lunesEntradaHoras     Hora de entrada del viajero el lunes al trabajo.
* @param lunesEntradaMinutos   Minuto de entrada del viajero el lunes al trabajo.
* @param lunesSalidaHoras      Hora de salida del viajero el lunes al trabajo.
* @param lunesSalidaMinutos    Minuto de salida del viajero el lunes al trabajo.
* @param martesLaboral         Indica si el martes es día laboral o no.
* @param martesEntradaHoras    Hora de entrada del viajero el martes al trabajo.
* @param martesEntradaMinutos  Minuto de entrada del viajero el martes al trabajo.
* @param martesSalidaHoras     Hora de salida del viajero el martes al trabajo.
* @param martesSalidaMinutos   Minuto de salida del viajero el martes al trabajo.
* @param miercolesLaboral      Indica si el miércoles es día laboral o no.
* @param miercolesEntradaHoras Hora de entrada del viajero el miércoles al trabajo.
* @param miercolesEntradaMinutos Minuto de entrada del viajero el miércoles al
    trabajo.
* @param miercolesSalidaHoras  Hora de salida del viajero el miércoles al trabajo.
* @param miercolesSalidaMinutos Minuto de salida del viajero el miércoles al trabajo.
* @param juevesLaboral         Indica si el jueves es día laboral o no.
* @param juevesEntradaHoras    Hora de entrada del viajero el jueves al trabajo.
* @param juevesEntradaMinutos  Minuto de entrada del viajero el jueves al trabajo.
* @param juevesSalidaHoras     Hora de salida del viajero el jueves al trabajo.
* @param juevesSalidaMinutos   Minuto de salida del viajero el jueves al trabajo.
* @param viernesLaboral        Indica si el viernes es día laboral o no.
```

* @param viernesEntradaHoras Hora de entrada del viajero el viernes al trabajo.

* @param viernesEntradaMinutos Minuto de entrada del viajero el viernes al trabajo.

* @param viernesSalidaHoras Hora de salida del viajero el viernes al trabajo.

* @param viernesSalidaMinutos Minuto de salida del viajero el viernes al trabajo.

* @param sabadoLaboral Indica si el sábado es día laboral o no.

* @param sabadoEntradaHoras Hora de entrada del viajero el sábado al trabajo.

* @param sabadoEntradaMinutos Minuto de entrada del viajero el sábado al trabajo.

* @param sabadoSalidaHoras Hora de salida del viajero el sábado al trabajo.

* @param sabadoSalidaMinutos Minuto de salida del viajero el sábado al trabajo.

* @param domingoLaboral Indica si el domingo es día laboral o no.

* @param domingoEntradaHoras Hora de entrada del viajero el domingo al trabajo.

* @param domingoEntradaMinutos Minuto de entrada del viajero el domingo al trabajo.

* @param domingoSalidaHoras Hora de salida del viajero el domingo al trabajo.

* @param domingoSalidaMinutos Minuto de salida del viajero el domingo al trabajo.

* @return Verdadero si el viajero se ha añadido correctamente y falso en caso contrario.

*/

2.2.5.12. HorarioViajero:

/**

* Modelo que almacenará la información del horario semanal de un viajero, es decir, las horas a las que entrará y saldrá cada día de la semana.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

- **EntradaSalida lunes:** Horario de entrada y salida correspondiente al lunes.
- **EntradaSalida martes:** Horario de entrada y salida correspondiente al martes.
  …
- **EntradaSalida domingo:** Horario de entrada y salida correspondiente al domingo.
- **HorarioViajero():**

  /**

  * Constructor vacío del horario.

  */

- **HorarioViajero(EntradaSalida lunes, EntradaSalida martes, EntradaSalida miercoles, EntradaSalida jueves, EntradaSalida viernes, EntradaSalida sabado, EntradaSalida domingo):**

  /**

  * Constructor que genera el horario correspondiente al viajero.

  *

  * @param lunes Horario del viajero el lunes.

  * @param martes Horario del viajero el martes.

  * @param miercoles Horario del viajero el miércoles.

  * @param jueves Horario del viajero el jueves.

  * @param viernes Horario del viajero el viernes.

  * @param sabado Horario del viajero el sábado.

  * @param domingo Horario del viajero el domingo.

  */

2.2.5.13. Viajero

/**

* Clase que almacenará la información de los viajeros.

* Los viajeros serán todos aquellos trabajadores implicados en "La Rueda" y se almacenará la información necesaia.

*

* @author Marcos Unzueta Puente @srunzu15 en github

* @version 1.0

*/

int idViajero: ID del viajero.

String nombre: Nombre del viajero.

String primerApellido: Primer apellido del viajero.

String segundoApellido: Segundo apellido del viajero.

HorarioViajero horario: Horario de trabajo del viajero.

- **Viajero(String nombre, String primerApellido, String segundoApellido, HorarioViajero horarioViajero):**

  /**

  * Constructor por defecto de la clase, que instanciará el objeto a partir de los valores.

  *

  * @param nombre Nombre del Viajero.

  * @param primerApellido Primer apellido del viajero.

* @param segundoApellido Segundo apellido del viajero.

*/

- **Viajero(int idViajero, String nombre, String primerApellido, String segundoApellido, HorarioViajero horario):**

  /**

  * Constructor que permite generar la instancia de Viajero a mano.

  *

  * @param idViajero Id del viajero.

  * @param nombre Nombre del Viajero.

  * @param primerApellido Primer apellido del viajero.

  * @param segundoApellido Segundo apellido del viajero.

  */

- **Viajero(Cursor cursor):**

  /**

  * Constructor que instanciará un objeto a partir de los datos obtenidos del cursor.

  *

  * @param cursor Cursor que ha de provenir de un select a la tabla Viajeros de la base de datos

  */

- **boolean almacenarEnLaBaseDeDatos(Activity controlador):**

  /**

  * Almacena el viajero en la base de datos, así como sus días de viajero.

  *

  * @param controlador Controlador que solicita almacenar el viajero.

  * @return Indica si la información se ha podido almacenar en la base de datos o no.

  */

- **boolean modificarEnLaBaseDeDatos(Activity controlador):**

  /**

  * Modifica el viajero en la base de datos, así como sus días de viajero.

  *

  * @param controlador Controlador que solicita modificar el viajero.

  * @return Indica si la información se ha podido modificar en la base de datos o no.

  */

- **String generarNombrePDF():**

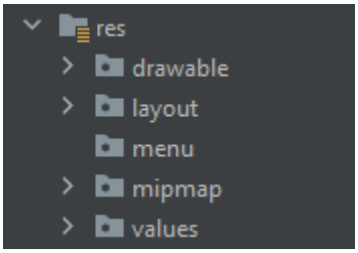
```
/**
* Genera el nombre de usuario que se mostrará en el PDF.
*
* @return Nombre único del viajero a mostrar en el PDF.
*/
```

2.3. Res

Esta carpeta se compone de archivos arbitrarios que se almacenan sin procesar. (android, s.f.)

Este directorio se dividirá en otras subcarpetas, que serán específicas de imágenes, menús, archivos XML, estilos y otros elementos que son utilizados en la aplicación por el resto de los ficheros.

Tabla 10: Esquema de res

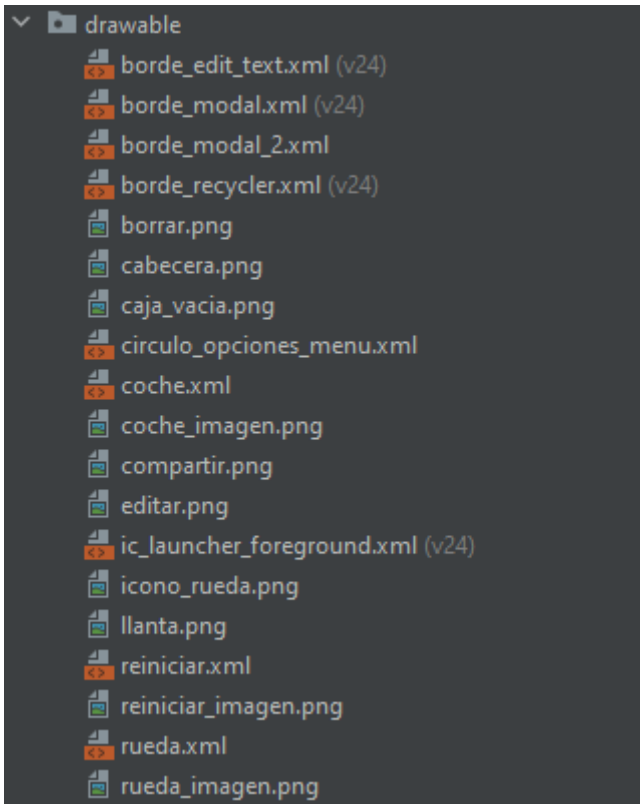


2.3.1. Drawable

Contiene elementos gráficos que se utilizan en la aplicación. (edu, s.f.)

En este caso se almacenan las imágenes, algunas construcciones xml utilizadas en el menú principal y bordes que se utilizan en algunas vistas de la aplicación como EditTexts, Modales o RecyclerViews.

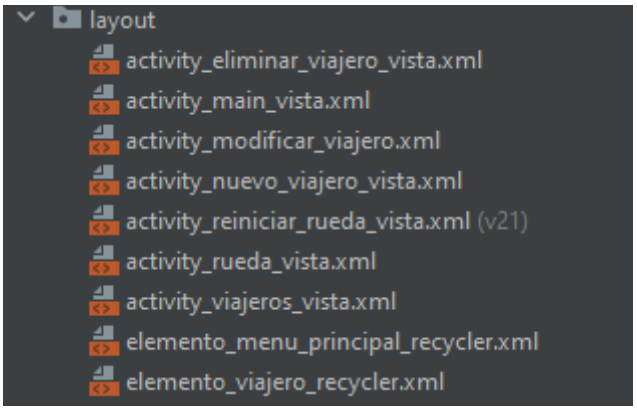
Tabla 11: Esquema de res.drawable



2.3.2. Layout

Contiene los ficheros XMl que harán las veces de vista en la arquitectura Modelo-Vista-Controlador y definen la interfaz del usuario (android, s.f.).

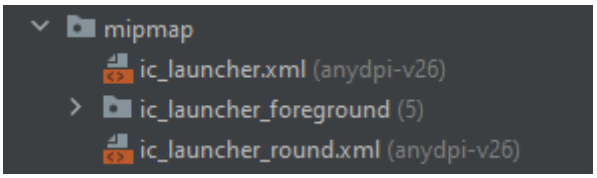
Tabla 12: Esquema de res.layout



2.3.3. Mipmap

Contiene los iconos de la aplicación en distintas resoluciones y tamaños. (android, s.f.)

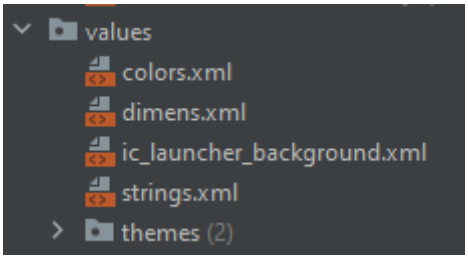
Tabla 13: Esquema de res.mipmap



2.3.4. Values

Archivos de tipo XML que contiene valores simples como cadenas de caracteres, enteros o colores.

Tabla 14: Esquema de res.values



2.3.4.1. colors.xml

XML que almacena los valores de color.

2.3.4.2. dimens.xml

XML que contiene los valores de dimensión.

2.3.4.3. strings.xml

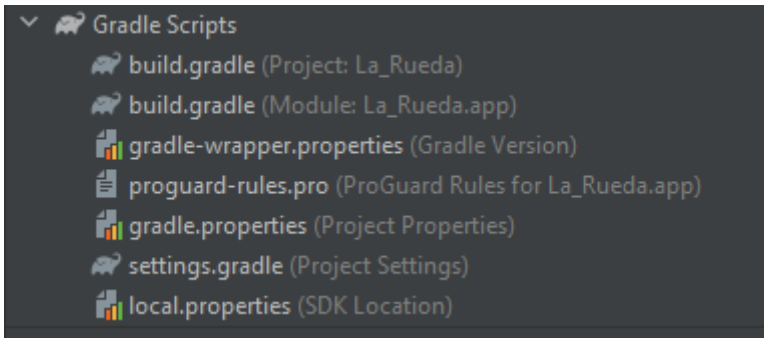
XML que contiene las cadenas de texto que se mostrarán en la aplicación.

A partir de este fichero se puede traducir la aplicación a otros idiomas, en este caso la aplicación se ha desarrollado en español.

2.4. Gradle

Contiene los datos que utilizará el proyecto para compilar, entre otras cosas las dependencias que importan las bibliotecas utilizadas, la mínima versión que soportará la aplicación o la versión de SDK de Android usada para compilar.

Tabla 15: Esquema de gradle



3. Referencias

Android. (s.f.). *https://developer.android.com*. Obtenido de https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419#:~:text=Todos%20los%20proyectos%20de%20apps,operativo%20Android%20y%20Google%20Play.

android, D. (s.f.). *https://developer.android.com*. Obtenido de https://developer.android.com/guide/topics/resources/providing-resources?hl=es-419

edu. (s.f.). *https://stuff.mit.edu/*. Obtenido de https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/resources/drawable-resource.html