5.1.5 Interfaz de líneas de bus

En este interfaz se podrá acceder a la información sobre las líneas de autobuses urbanos de la ciudad, tanto las paradas de cada línea como las frecuencias de las distintas líneas según la hora del día.

A este interfaz se accede al pulsar el botón cuyo icono es la vista frontal de un autobús en la barra de herramientas (formulario *Tools*). En la figura 5.1.5.1 se muestra el aspecto de este interfaz.

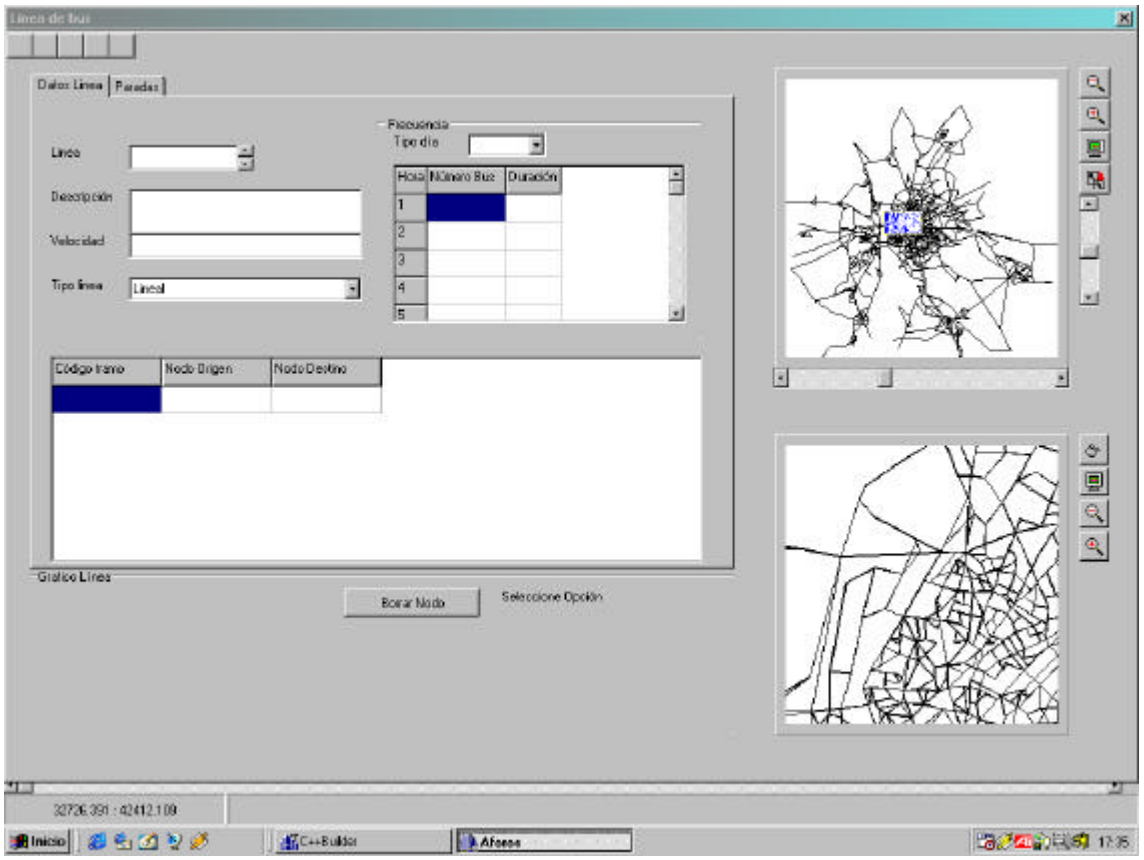

Figura 5.1.5.1: Vista del interfaz de líneas de autobús

Puede apreciarse como este interfaz también se divide en dos páginas. La primera de ellas, que se muestra en la figura 5.1.5.1, es la página de líneas. En ella se representan los datos de cada una de las líneas.

Puede usarse de dos formas, para ver información y para añadirla.

Si se teclea el número de una línea en el cuadro de diálogo línea, los demás cuadros de diálogo mostrarán la información correspondiente a esa línea: la descripción de la línea, la velocidad y el tipo de línea, que puede ser lineal, circular o mixta.

Abajo aparece una estructura similar a una matriz en la que se muestra los diferentes elementos de la red vial que recorre la línea. Por ejemplo, si la línea comienza en el nodo 50, y sigue luego hasta el 60 y luego el 70, esa matriz contendrá dos filas. La primera de ellas tendrá en nodo origen 50 y en nodo destino 60. El código de tramo indicará además el tramo que une a esos dos nodos. La segunda fila contendrá como

nodo origen 60 y como destino 70. De nuevo el código tramo de esta fila mostrará el código del tramo que une los nodos 60 y 70.

En el marco inferior, etiquetado como Gráfico Línea, se mostrará gráficamente de la línea, representando los nodos como círculos y los tramos como rectas uniendo dichos círculos. Una vez cargada la información de la línea, podrán borrarse nodos de la misma pulsando sobre el nodo a eliminar en la representación gráfica de la línea, y pulsando a continuación el botón Borrar Nodo. Esto borra el nodo y el tramo que lo conectaba con el nodo anterior.

También se pueden añadir nuevos nodos pulsando el botón cuyo icono representa una mano con un dedo apuntando, junto a la imagen inferior de las dos qua aparecen de la red vial. Tras pulsar este botón, si se selecciona un nodo de la red vial sobre la imagen inferior, este nodo queda añadido a la línea si es adyacente al último nodo añadido. Los tramos se añaden automáticamente al añadir nodos conectados entre sí. Si los nodos no están conectados no se puede añadir el nuevo nodo.

Las dos imágenes que aparecen en la parte derecha se ha creado y se comportan de igual manera que en los demás interfaces, la única diferencia es la función que se realiza al seleccionar un nodo, que en este caso produce su adición a la línea de autobús.

A la derecha de los cuadros de diálogo usados para presentar la información de la línea hay un marco con la leyenda Frecuencia. Abajo aparece un cuadro de diálogo, tipo de dia, y una matriz con los datos Hora, Número de bus y Duración.

La función de este marco es representar la duración del recorrido de los autobuses según el día (tipo de día) y las horas del día. Por todos es sabido que la frecuencia con la que circulan los autobuses varía dependiendo del día (domingo o lunes) y de la hora (8:00 frente a10:00, por ejemplo).

Si el usuario pincha sobre la etiqueta de Parada, se despliega la página de paradas, como se muestra en la figura 5.1.5.2:

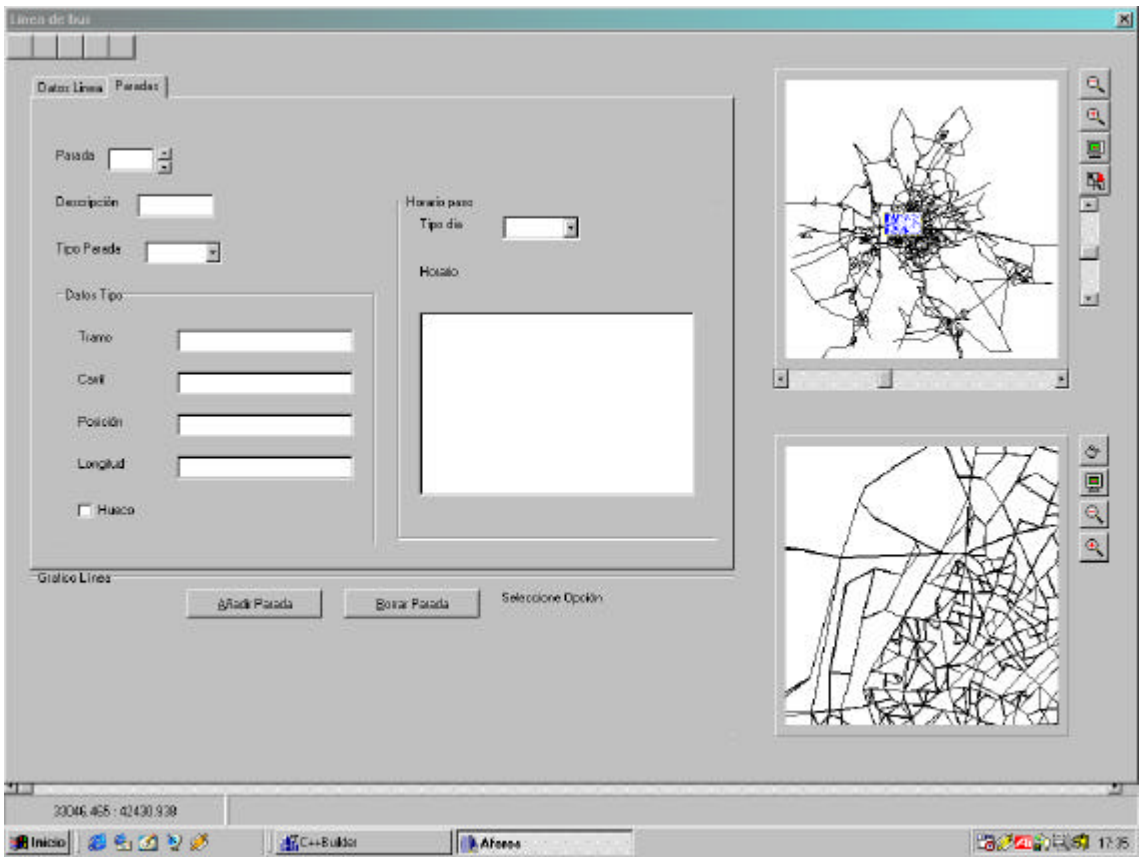

Figura 5.1.5.2: Página de paradas

En esta página se accede a la información sobre las paradas de la línea. Se muestra una serie de información general sobre cada parada, como una descripción de la misma, el tipo de parada que es, el tramo y carril dentro del tramo donde se encuentra, la posición desde el origen del tramo y la longitud de la parada.

En la parte derecha aparecerá información sobre las frecuencias de paso en función del tipo de día y de la franja horaria, al igual que en la página anterior.

Y también aparece el gráfico de la línea, pero en esta ocasión mostrando las paradas. En este gráfico se puede seleccionar uno cualquiera de los tramos que forman la línea y añadir una parada en ese tramo o eliminarla.

Implementación del interfaz de líneas de bus

```
class Posicion_bus
{
 unsigned int codigo_nodo;
 int x;
 int y;
 public:
// métodos públicos de la clase
};
```

En realidad esta clase es auxiliar, sólo se usa como ayuda para representar los nodos en la gráfica de la línea, la que aparece en la parte inferior de la pantalla.

De hecho, las propiedades que tiene son *codigo_nodo*, que indica el nodo al que representa y *x* e *y*, que son las coordenadas de dicho nodo en la imagen.

```
class Parada
{
 bool hueco;
 unsigned int cod_parada;
 unsigned int cod_linea;
 unsigned int tipo_parada;
 unsigned int codigo; // de tramo o de nodo
 unsigned int codigo_sub; // de movimiento o de carril
 float posicion;
 float longitud;
 char descripcion[30];
 bool tramo;
public:
//métodos públicos de la clase
};
```

Es la estructura de datos básica para almacenar información sobre la parada. De hecho todos los métodos que tiene se usan para devolver o actualizar valores de variables.

La propiedad *cod_parada* se usa como identificador único de la parada. La propiedad *cod_linea* sirve para identificar a la línea que pertenece la parada.

La propiedad *tipo_parada* indica el tipo de parada de que se trata. La propiedad *codigo* indica el tramo o el nodo donde está la parada. La propiedad *codigo_sub* indica el codigo de movimiento o carril en el que se encuentra.

La propiedad *posicion* indica la posicion en la que se encuentra la parada, medida desde el comienzo del tramo, si es que está emplazada en un tramo. La propiedad *longitud* indica la longitud de la parada. La propiedad *descripcion* es una descripción de la misma.

La propiedad *tramo* indica si la parada tiene lugar en un tramo o en un nodo, y por lo tanto indica si la propiedad *codigo* es un codigo de un tramo o de un nodo. La propiedad *hueco* indica si existe un hueco en la calzada para la parada.

```
class Parada_interactiva
{
  TTabSheet *parada;
  TLabel *l_descripcion;
  TEdit *e_descripcion;
  TLabel *l_parada;
  TEdit *e_parada;
  TUpDown *u_parada;
  TLabel *l_tipo;
  TComboBox *e_tipo;
  TGroupBox *GroupBox1;
```

```
  TGroupBox *GroupBox2;
  TLabel *l_codigo1;
  TEdit *e_codigo1;
  TLabel *l_codigo2;
  TEdit *e_codigo2;
  TLabel *l_posicion;
  TEdit *e_posicion;
  TLabel *l_longitud;
  TEdit *e_longitud;
  TCheckBox *hueco;
  TLabel *l_tipo_dia;
  TComboBox *tipo_dia;
  TLabel *l_horario;
  TListBox *horario;
  bool Ins_parada;
  bool Bor_parada;
public:
   void Eventos_linea (TImage *imagen);
   void Generar_datos_parada (TWinControl *padre,int Ancho_p,int Alto_p);
   void Destruir_formulario ();
   void __fastcall Sel_nodo (TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y);
   void Eventos_parada (TImage *imagen);
   void __fastcall Insertar_parada (TObject *Sender, TMouseButton Button, TShiftState
Shift, int X, int Y);
   void Eventos_borrar_nodo (TImage *imagen);
};
```

La función principal de esta clase es representar la página de paradas y asignar los eventos correctamente.

El método *Generar_datos_parada* lo que hace es crear los objetos gráficos para formar la pagina de paradas.

El método *Eventos_parada* asigna el método *Insertar_Parada* al evento *onMouseUp* sobre la imagen donde está representada la línea.

El método *Eventos_linea* asigna el evento *onMouseUp* sobre la imagen que representa las líneas al método *Sel_nodo.*

El método *Seleccionar* pone los indicadores *Ins_parada* y *Bor_parada* a *false*.

El método *Insertar_parada* añade una parada: la dibuja en el diagrama que representa la línea y marca el tramo o nodo correspondiente de la lista *Lista_tramos_parada*. Esto se hace poniendo la propiedad *para* de dicho objeto a *true*.

El método *Eventos_borrar_nodo* borra la parada de la lista de paradas, poniendo la propiedad *para* a *false.*

El método *Borrar* pone los indicadores *Bor_parada* a *true* e *Ins_parada* a *false*.

El método *Destruir_formulario* libera la memoria de todos los objetos gráficos de la página de parada.

El método *Sel_nodo* determina el nodo seleccionado en la imagen y según los valores de los indicadores *Bor_parada* e *Ins_parada* realiza una acción u otra. Si *Ins_parada* está activado, llama a *Insertar_parada*, si es *Bor_parada* el que está activado, llama a *Eventos_borrar_nodo*. Si ninguno de los dos está activado, muestra información sobre la parada seleccionada.

```
class Datos_linea
{
 TTabSheet *linea;
 TUpDown *u_linea;
 TLabel *l_linea;
 TLabel *l_descripcion;
 TLabel *l_velocidad;
 TLabel *l_tipo_linea;
 TComboBox *tipo_linea;
 TEdit *e_linea;
 TEdit *e_descripcion;
 TEdit *e_velocidad;
 TCheckBox *activada;
 TGroupBox *GroupBox1;
 TLabel *l_tipo_dia;
 TComboBox *tipo_dia;
 TStringGrid *grid;
 TStringGrid *tramos_linea;
public:
   void Generar_datos_linea (TWinControl *padre,int Ancho_p,int Alto_p);
   void Destruir_formulario();
  };
```

El método *Generar_datos_linea* tiene como función construir la página de líneas, creando todos los objetos gráficos necesarios.

El método *Destruir_formulario* libera la memoria reservada para todos los objetos gráficos.

```
class Linea_bus_interactivo
{
TPageControl *pagina;
 TImage *Grafico_linea;
 TGroupBox *GroupBox1;
 TForm *formulario;
 TBevel *caja1,*caja2;
 Parada_interactiva parada;
 Datos_linea datos;
 TBitBtn *Add_nodo;
 TBitBtn *Borrar_nod;
```

```
 TLabel *l_informacion;
 TToolBar *barra;
 Lista_inf<TToolButton*> botones_barra;
public:
  void Generar_datos_linea (TWinControl *padre);
  void __fastcall Evento_sel_nodo (TObject *Sender);
  void __fastcall Evento_pintar_linea (TObject *Sender);
  void __fastcall Sel_nodo (TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y);
  void Activacion_seleccion_planes (Contexto *context);
  void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
  void __fastcall Cambio(TObject *Sender);
  void Destruir_formulario();
  void Actualiza_eventos();
  void __fastcall Evento_Borrar_nodo (TObject *Sender);
  void Buscar_nodo_tramo (int X,int Y, unsigned int *i1,unsigned int *i2);
  bool Validez_nodo(unsigned int codigo1);
  void Representa_linea ();
};
```

Esta clase se encarga de dibujar el interfaz y de asignar los distintos eventos a los métodos adecuados.

Todas las propiedades son objetos de representación gráfica a excepción de *parada* y *datos*. No obstante, estos objetos tienen como función representar la página de líneas y de paradas.

El primer método al que se llama para crear el interfaz de líneas de autobús es a *Generar_datos_linea* de la clase *Linea_bus_interactivo.*

Este método lo primero que hace es crear las cuatro listas que se usarán para almacenamiento de datos, es decir, *Lista_lineas, Lista_tramos_parada, Horarios* y *Paradas_frec*.

A continuación crea la parte del interfaz que es común a las dos páginas. Es decir, crea el controlador de páginas, la imagen inferior donde se representa la línea o las paradas y los botones correspondientes. Crea todos los botones, después según la página en la que se esté, estarán visibles unos u otros.

Asigna el evento *onChange* sobre el controlador de página al método *Cambio*. Llama al método *Generar_datos_linea* de su propiedad *datos* (clase *Datos_linea*) y al método *Generar_datos_parada* de su propiedad *parada* (clase *Parada_interactiva*).

Crea dos botones, *Add_nodo* y *Borrar_nod*. En principio *Add_nodo* no es visible, sólo *Borrar_nod*, con la leyenda *Borrar Nodo*. Cuando el usuario seleccione la página de paradas, se cambiarán las leyendas de estos botones, que aparecerán como *Añadir Parada* y *Borrar Parada*, se hará visible el botón *Add_nodo* y se le asignarán métodos diferentes al evento *onClick* sobre cada uno de ellos. Es decir, los mismos botones son usados por las dos páginas.

Por defecto se presenta la página de líneas, por lo que se asignan los siguientes métodos a los botones: el método *Evento_pintar_linea* al evento *onClick* sobre el botón *Add_nodo*, y el método *Evento_Borrar_nodo* al botón *Borrar_nod*. Aunque el botón *Add_nod* no estará disponible en ese momento, se queda preparado para actuar con la página de paradas, pues llama pone el indicador *Ins_parada* del objeto parada a *true* y llama al método *Eventos_parada* del objeto parada.

También llama al método *Pinta_mapa* de la clase *Informacion_pantalla*, para representar las dos imágenes de la parte derecha del interfaz que muestran la red vial, de manera que el evento onMouseUp sobre la imagen inferior que representa la red vial queda asignado al método *Evento_sel_nodo*.

También asigna los eventos llamando al método *Actualiza_eventos*. Finalmente activa los objetos *contexto* y *contexto_giro* (método *Activar* de la clase *Contexto*).

El método *Cambio* realiza dos secuencias de acciones según la página que haya seleccionado el usuario. Si la página seleccionada es la de líneas, llama al método *Eventos_linea* de la clase *Parada_interactiva,* oculta el botón *Add_nodo* y hace que el botón *Borrar_nod* muestre la leyenda *Borrar_nodo*.

Si por el contrario la página seleccionada es la de paradas, hace visible el botón *Add_nod* y cambia las leyendas de los botones a *Añadir_Parada* para *Add_nodo* y a *Borrar_Parada* para *Borrar_nod*. Finalmente llama a los métodos *Eventos_parada* y *Seleccionar* del objeto *parada*.

El método *Eventos_parada* es análogo a *Eventos_línea* pero el método que asigna es *Insertar_parada* de la clase *Parada_interactiva*.

El método *Seleccionar* lo que hace es poner a *false* los indicadores *Ins_parada* y *Bor_parada* del objeto *parada*.

El método *Evento_sel_nodo* asigna los siguientes eventos sobre la imagen inferior de las que representan la red vial a los siguientes métodos: el evento *onMouseMove* lo asigna al método *Desplazamiento_raton_imagen* de la clase *Contexto,* ya conocido. El evento *onMouseDown* lo asigna al método *Botón_abajo_nulo* de la clase *Contexto,* también conocido. Finalmente el evento *onMouseUp* queda asignado al método *Sel_nodo* de la clase *Linea_bus_interactivo.*

El método *Sel_nodo* lo primero que hacer es convertir el punto sobre el que se ha pinchado en la imagen a coordenadas globales para obtener el punto que el usuario ha querido seleccionar. A continuación busca el último nodo de la línea de autobús cuyo número aparece en el cuadro de diálogo etiquetado como línea. Esta será la línea de autobús seleccionada. En dicha imagen se representa adecuadamente el punto seleccionado, es decir, se representa mediante el ya conocido círculo inscrito en un cuadrado. Busca si la línea de autobús existe ya. Si no existe crea el objeto *Linea_bus_inf* y lo añade a la lista *Lista_lineas*, y a continuación crea el elemento *tramo_Parada* y lo añade a la *Lista_tramos_parada* adecuadamente, actualizando el puntero en la lista *Lista_lineas.* En caso de existir, se crean dos objetos *tramo_Parada*, el primero de ellos contiene la información sobre el tramo que une al último nodo de la línea con el seleccionado. El segundo contiene la información del nodo seleccionado.

Añade estos dos elementos adecuadamente en la lista *Lista_tramos_parada*, es decir, actualizando el puntero *s_tramo* y el indicador *ultimo* de los elementos anteriores de la misma línea.

Finalmente llama al método *Representa_línea* que se encarga de representar los nodos y tramos de la línea en la imagen de la parte inferior de la pantalla, *Grafico_linea*.

El método *Actualiza_eventos* asigna el evento *onClose* sobre la propiedad *formulario* al método *FormClose*. Es decir, cuando se cierre el interfaz de líneas de autobús se llamará al método *FormClose*. Este método libera la memoria usada por todas las listas que se han creado en este interfaz, y libera los objetos gráficos creados (usando el método *Destruir_formulario*) y hace visible la barra de herramientas.

El método *Destruir_formulario* destruye los objetos gráficos comunes de ambas páginas y llama a los métodos *Destruir_formulario* de los objetos *datos* y *parada*, para destruir cada uno de los objetos gráficos de las páginas de líneas y paradas. También desactiva los objetos *contexto* y *contexto_giro*.

El método *Evento_Borrar_nodo* actúa de forma distinta según la página en que se encuentre el usuario. Si está en la página de líneas llama al método *Eventos_borrar_nodo* del objeto parada. En caso contrario llama al método *Borrar* del mismo objeto.

```
class tramo_Parada
{
Parada p;          // Datos de la parada
bool para;    // ¿ Para realmente aqui? ¿ o solo pasa por aqui?
unsigned int s_tramo; // sigte. tramo dentro de la misma linea
unsigned int a_tramo; // anterior tramo dentro de la misma linea
bool con_ant; // ¿ tiene conectividad con el anterior?
bool con_post; // ¿tiene continuidad con el posterior?
bool ultimo; // ¿ Tiene algun elemento detras?
bool primero; // ¿Tiene algun elemento delante?
public:
tramo_Parada (Parada, bool);
tramo_Parada ();
};
```

Esta clase almacena información sobre cada uno de los nodos o tramos por los que circula cada una de las líneas de autobús. En la propiedad *p*, de la clase *Parada*, se almacenan todos los datos relevantes sobre si es un tramo, un nodo, etcétera.

La propiedad *para* indica si hay una parada en este tramo o nodo o simplemente la línea pasa por aquí. La propiedad *s_tramo* es un puntero al siguiente elemento de la lista Lista_tramos_parada por la que pasa la misma línea. La propiedad *a_tramo* es análoga pero el elemento anterior. Esto hace que si se desea borra una parada de una línea no hay que modificar toda la estructura, simplemente marcar *para* como *false*.

Las propiedades *cont_ant* y *cont_post* indican si se tiene conectividad con el elemento anterior o posterior de la línea de autobús: aunque los elementos estén enlazados en la

lista mediante los punteros *s_tramo* y *a_tramo*, es posible que físicamente esos elementos de la red vial no estén unidos. Esta situación se da cuando el usuario borra un nodo intermedio de la línea. Sin embargo, se mantiene esa información por si el usuario quiere crear un camino alternativo que una los elementos actualmente desconectados y restuare así la línea.

La propiedad *ultimo* indica si es el ultimo de los elementos para una línea, que no de la lista, y la propiedad *primero* indica si el primer elemento de una línea, que no de la lista.

Puede observarse como los métodos más importantes son los constructores. Y hay dos. El que no recibe parámetros crea un objeto auxiliar, que se usará como ayuda a la programación. Sin embargo, el que recibe parámetros se encarga de crear un objeto con la información necesaria y de añadirlo a la lista *Lista_tramos_parada*.

La lista *Lista_tramos_parada* es de la clase *Lista_inf* y sus elementos son objetos de la clase *Tramos_Parada*.

Esta lista contendrá la información de todas las líneas. En efecto, cada vez que se añade un nuevo elemento (nodo o tramo) se añade a la lista *Lista_tramos_Parada*, sin importar la línea a la que pertenezca. Cada vez que se añade un elemento a esta lista se comprueba si hay ya otros elementos y se actualizan los campos *s_tramo* y *a_tramo* adecuadamente, de manera que apunten al elemento (nodo o tramo) anterior en la lista y al siguiente (cuando se añada, posteriormente) de manera que es como si se tuviese una lista creada mediante punteros para cada línea dentro de la lista lógica *Lista_tramos_parada*. Y es que esto aporta una ventaja fundamental, puesto que siempre hay que leer el mismo fichero para restaurar los datos de todas las líneas, independientemente de las líneas que se tengan.

Como se verá posteriormente, existe otra lista que almacena las diferentes líneas que existen. En esta lista se encuentra, para cada línea, la posición del primer elemento que pertenece a dicha línea en la lista *Lista_tramos_parada*. Esto permite saber fácilmente los elementos (nodos o tramos) que componen una línea, ya que obtenido el primer elemento, sólo hay que seguir los punteros *s_tramo* y *a_tramo* para obtener el siguiente elemento de la línea o el posterior, respectivamente, lográndose la misma rapidez que si se hubiese almacenado cada línea en un fichero separado, pero sin los incovenientes de esta última solución.

```
class Linea_bus_inf
{
unsigned int codigo_linea;
unsigned long comienzo; //puntero a lista de paradas (Lista_tramos_parada)
AnsiString descripcion; // Descripcion de la linea
float velocidad;
unsigned short tipo_linea; //0- lineal; 1- circula; 2- mixta
public:
Linea_bus_inf (unsigned int);
Linea_bus_inf ();
};
```

Esta clase contiene, para la línea cuyo código se indica en la propiedad *codigo_linea,* la posición del primer elemento (nodo) de dicha línea en la lista *Lista_tramos_parada*. Esta posición es almacenada por la propiedad *comienzo*.

Las demás propiedades proporcionan información de tipo general sobre la línea: *descripcion, velocidad* y *tipo_linea*. El tipo de linea puede ser lineal, circular o mixta.

Existe una lista de tipo *Lista_inf*, llamada *Lista_lineas,* que contiene objetos de la clase *Linea_bus_inf*. De esta manera, cada vez que se añade una línea nueva basta con añadir un elemento a esta lista. Y para borrarla se borran todos los elementos de dicha línea en *Lista_tramos_parada* (muy fácil siguiendo los punteros) y se borra el elemento correspondiente de *Lista_lineas.*

De nuevo los únicos métodos interesantes son los constructores, que están sobrecargados de nuevo. Al igual que en el caso anterior, la variante sin parámetros sirve para crear objetos auxiliares y la variante con parámetros se crea con el fin de almacenar información útil y que sea añadido a la lista.

```
class Parada_f
{
unsigned long cod_lin;
unsigned long codpar;
unsigned short frec; //frecuencia
unsigned long sig; // puntero a siguiente elto de la misma linea
public:
};
```

Esta clase sirve para almacenar la frecuencia (propiedad *frec*) con la que pasa un autobús perteneciente a cierta línea (propiedad *cod_lin*) por cierta parada (propiedad *cod_par*).

La propiedad *sig* indica la posición dentro de la lista *Paradas_frec* donde se encuentra el siguiente elemento que contiene información sobre esta misma franja horaria. Esto se entenderá mejor en unos instantes.

```
class horario
{
unsigned long comienzo;//puntero a elto dentro de la lista de Parada_f para
unsigned short franja; //franja horaria sera un numero
public:
// métodos públicos de la clase
};
```

La propiedad *franja* indica la hora del dia a la que se refiere una cierta frecuencia, como se verá. La propiedad *comienzo* contiene la posición en la lista *Paradas_frec* del primer elemento que guarda información sobre la frecuencia de paso de cierta línea y cierta parada en la frecuencia horaria indicada.

Los elementos de esta clase sirven para formar una lista de tipo *Lista_inf* llamada *Horarios.* En esta lista se almacenan las distintas franjas horarias existentes, y permite la búsqueda en la lista *Paradas_frec* de las frecuencias de las distintas líneas y paradas para cada franja horaria, análogamente a la estructura que siguen las listas *Lista_tramos_parada* y *Lista_lineas*.

Tanto en esta clase como en la anterior, los constructores tienen funciones análogas a los constructores de las clases *Linea_bus_inf* y *Tramos_Parada*, respectivamente.

En la figura siguiente, figura 5.1.5.3, se muestra un esquema de la estructura usada por este interfaz para almacenar la información. Las líneas sólidas unen los objetos que pertenecen a la misma lista, y las líneas discontinuas muestran como un objeto contiene un puntero a otro.

Puede apreciarse claramente como los elementos que forman la línea de bus número 1 son actualmente, los que ocupan las posiciones 0,2,4 y 6 en la lista *Lista_tramos_parada*. Los que forman la línea 2 son los elementos que ocupan las posiciones 1,3 y 7 y que la línea N tiene un solo elemento, el que ocupa la posición 5.

La estructura de las listas *Paradas_frec* y *Horarios* es análoga, salvo por el detalle de que no existe puntero al elemento anterior.

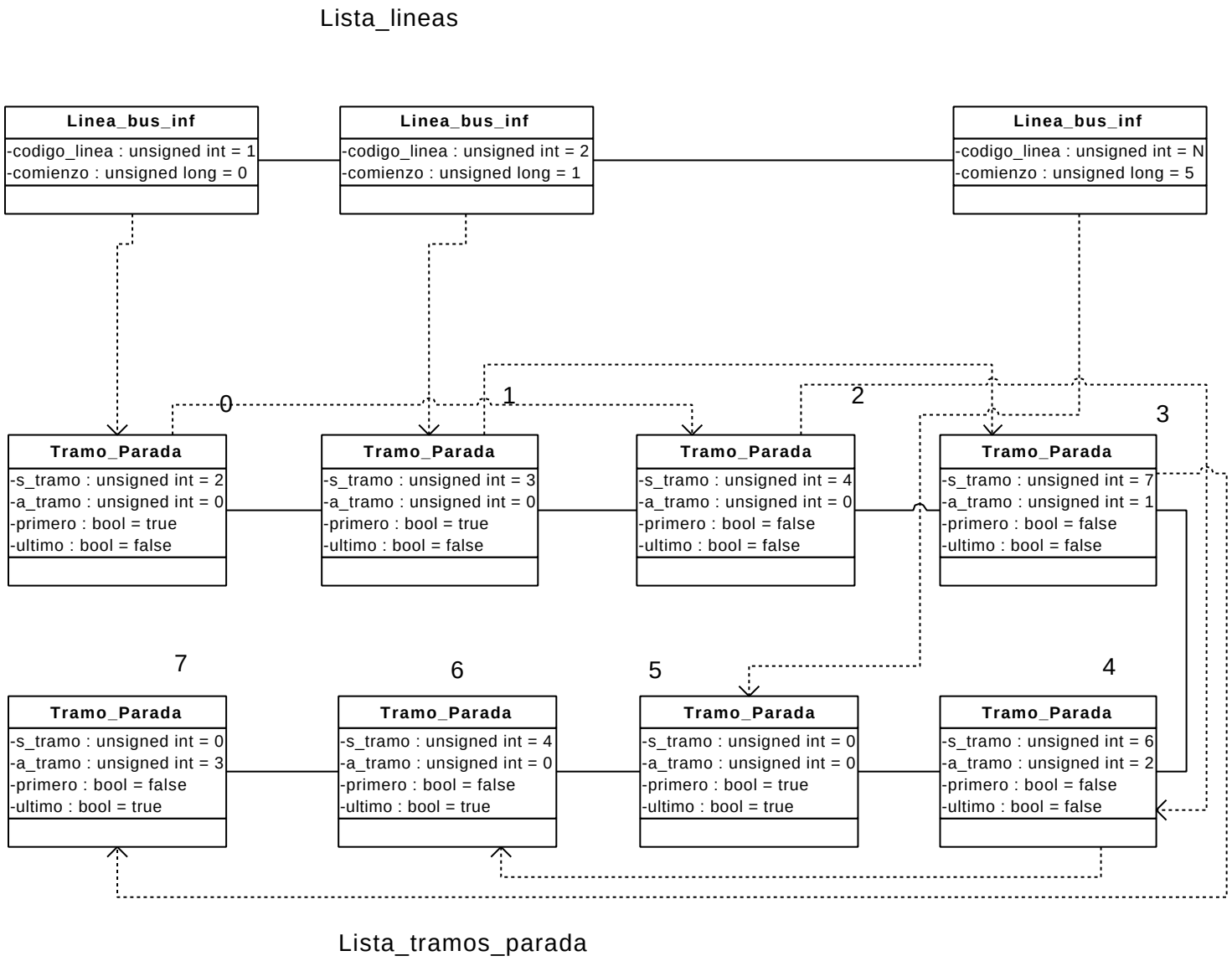

Figura 5.1.5.3: La estructura de la listas *Lista_tramos_parada* y *Lista_lineas*.