ANEXO E. MEJORAS Y AMPLIACIONES: CODIFICACIÓN DE PARÁMETROS.

A continuación se muestra una forma de estructurar y codificar los parámetros correspondientes a las recomendaciones que el servidor de recomendaciones ofrece en función de los parámetros que el usuario final le envíe.

Según lo definido, una recomendación estaría compuesta por diferentes alternativas para alcanzar un destino fijo, a partir de un origen también fijo. Dichas alternativas las denominamos itinerarios. A su vez cada itinerario estará compuesto por diferentes trayectos, de forma que un trayecto se define como el recorrido desde un punto geográfico hasta otro siempre usando un mismo modo de trasporte. Por su parte, cada trayecto quedará definido por una serie de parámetros que se muestran en el código fuente.

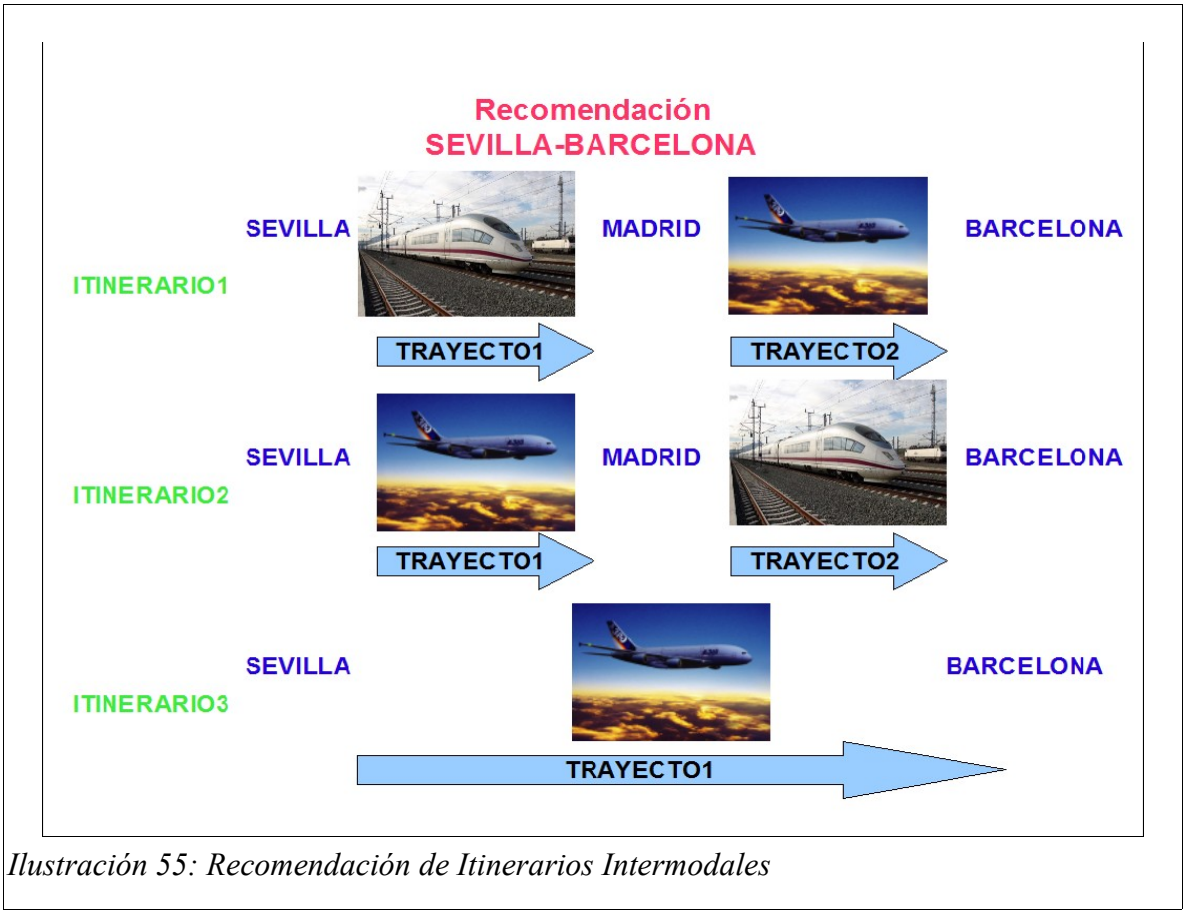

Ilustración 55: Recomendación de Itinerarios Intermodales

A continuación se muestra el código propuesto para las recomendaciones:

5;;RECOMENDACION1;;5;;0;t_time_S;;TRAYECTO1;;1;;t_time_LL;;100;;1;;COMPAÑIA1;;TRAYECTO2;;25;;t_time_LL;;105;;2;;COMPAÑIA2;;TRAYECTO3;;46;;t_time_LL;;77;;3;;COMPAÑIA3;;TRAYECTO4;;45;;t_time_LL;;145;;4;;COMPAÑIA4;;TRAYECTO5;;33;;t_time_LL;;110;;2;;COMPAÑIA5;;RECOMENDACION2;;3;;0;t_time_S;;TRAYECTO1;;1;;t_time_LL;;100;;1;;COMPAÑIA1;;TRAYECTO2;;25;;t_time_LL;;105;;2;;COMPAÑIA2;;TRAYECTO3;;46;;t_time_LL;;77;;3;;COMPAÑIA3;;RECOMENDACION3;;1;;0;t_time_S;;TRAYECTO1;;1;;t_time_LL;;100;;1;;COMPAÑIA1;;RECOMENDACION4;;5;;0;t_time_S;;TRAYECTO1;;1;;t_time_LL;;100;;1;;COMPAÑIA1;;TRAYECTO2;;25;;t_time_LL;;105;;2;;COMPAÑIA2;;TRAYECTO3;;46;;t_time_LL;;77;;3;;COMPAÑIA3;;TRAYECTO4;;45;;t_time_LL;;145;;4;;COMPAÑIA4;;TRAYECTO5;;33;;t_time_LL;;110;;2;;COMPAÑIA5;;RECOMENDACION5;;3;;0;t_time_S;;TRAYECTO1;;1;;t_time_LL;;100;;1;;COMPAÑIA1;;TRAYECTO2;;25;;t_time_LL;;105;;2;;COMPAÑIA2;;TRAYECTO3;;46;;t_time_LL;;77;;3;;COMPAÑIA3;;

Este sería el código que el servidor de recomendaciones ofrecería como resultado a nuestra aplicación SERVIDOR. Posteriormente, SERVIDOR la interpretaría y almacenaría cada parámetro en su correspondiente nodo. Esto es así porque nuestra aplicación va a almacenar esta información en una lista dinámica formada por nodos itinerario y nodos trayecto, cada uno con sus correspondientes parámetros. Una vez que SERVIDOR ha completado la lista, esta es enviada a la aplicación CLIENTE.

La interpretación del código sería la siguiente:

- El primer parámetro indica el número de itinerarios ofrecidos por el servidor de recomendaciones, que en este caso es 5.

- El sergundo es el identificador de recomendación que se asocia a un itinerario.
- El tercero es el número de trayectos que compone ese itinerario recomendado, que para la RECOMENDACIÓN1 es 5.
- El siguiente es la ciudad origen, que en este caso se identifica con un 0. El código de las ciudades se corresponde con el impuesto en el formulario web de petición de recomendación.
- El quinto es la hora de salida, t_time_s.
- El sexto es el identificador de trayecto, el cual puede ser útil a la hora de modificar en tiempo real algún trayecto. TRAYECTO1.
- El siguiente parámetro es el identificador de la ciudad destino. En este caso el destino del TRAYECTO1 es 1.
- El octavo parámtro es la fecha de llegada, t_time_ll.
- El noveno es el precio del viaje asociado al trayecto, 100 euros.
- El décimo es el modo de trasporte: 1 avion, 2 barco, 3 bus y 4 tren.
- El último es el identificador asociado a la compañía que gestiona el viaje de dicho trayecto.

El código fuente es el siguiente:

```
#include <stdio.h>
#include <string.h>

void extrae_parámetros_recomendacion(eltorecom** p, eltorecom** f, sms*
capsula_smsmo)
```

```
{
	char buffer[TAM_BUFFER];
	char* parametro = NULL;
	char* aux;
	itinerario* a;		// Puntero auxiliar para operar sobre el
itinerario
	eltotrayecto* b		// Puntero auxiliar para operar sobre el trayecto
	eltotrayecto* ini_trayecto;
	eltotrayecto* fin_trayecto;
	int primer_trayecto = 1 // Sirve para indicar si se trata del primer
trayecto de un itinerario
	int recomendaciones = 0;  // Numero de recomendaciones
	int trayectos = 0;		// Numero de trayectos
	int num_caracteres = 0; // Indica cuantos caracteres hay que avanzar para
el siguiente parametro


	while( !(pop_msj(p,f,buffer)) )
	{
		aux = buffer;

		// Obtengo el primer parametro que es el numero de recomendaciones
		parametro = extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);

		recomendaciones = atoi(parametro);

		num_caracteres = strlen(parametro);
		free(parametro);

		while(recomendaciones > 0)
		{
			aux = num_caracteres + 2;
			// Ahora aux apunta al identificador de itinerario

			parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
			num_caracteres = strlen(parametro);

			// Creamos un nodo itinerario y le asignamos el identificador
			push_itinerario(p,f,parametro,capsula_smsmo);
			free(parametro);

			a = *f;	// a apunta al Ã°ltimo nodo de la lista de
itinerarios
```

```
                aux = num_caracteres + 2;
                // Ahora aux apunta al nÃºmero de trayectos

                parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                num_caracteres = strlen(parametro);

                trayectos = atoi(parametro);

                // Actualizo el nodo itinerario
                a->num_trayectos = trayectos;

                free(parametro);

                aux = num_caracteres + 2;
                // Ahora aux apunta al origen

                parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                num_caracteres = strlen(parametro);

                // Actualizo el nodo itinerario
                a->origen = atoi(parametro);

                free(parametro);

                aux = num_caracteres + 2;
                // Ahora aux apunta a la hora de salida

                parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                num_caracteres = strlen(parametro);

                // Actualizo el nodo itinerario
                a->h_salida = atol(parametro);

                free(parametro);

                while(trayectos > 0)
                {
                      aux = num_caracteres + 2;
                      // Ahora aux apunta al identificador del trayecto

                      parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
```

```
                        num_caracteres = strlen(parametro);

                        // Creamos un nodo trayecto y le asignamos su
identificador

      push_trayecto(&ini_trayecto,&fin_trayecto,parametro,capsula_smsmo);
                        free(parametro);

                        // b apunta al Ã°ltimo nodo de la lista de trayectos
                        b = fin_trayecto;
                        a->ultimo = b;

                        aux = num_caracteres + 2;
                        // Ahora aux apunta al destino

                        parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                        num_caracteres = strlen(parametro);

                        b->destino = atoi(parametro);

                        free(parametro);

                        aux = num_caracteres + 2;
                        // Ahora aux apunta a la hora de llegada

                        parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                        num_caracteres = strlen(parametro);

                        b->h_llegada = atol(parametro);

                        free(parametro);

                        aux = num_caracteres + 2;
                        // Ahora aux apunta al precio

                        parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                        num_caracteres = strlen(parametro);

                        strcpy(b->precio,parametro);

                        free(parametro);
```

```
                        aux = num_caracteres + 2;
                        // Ahora aux apunta al modo

                        parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                        num_caracteres = strlen(parametro);

                        b->modo = atoi(parametro);

                        free(parametro);

                        aux = num_caracteres + 2;
                        // Ahora aux apunta a la compañía

                        parametro =
extrae_carecteres_hasta_marca(MARCA,aux,capsula_smsmo);
                        num_caracteres = strlen(parametro);

                        strcpy(b->compañía,parametro);

                        free(parametro);

                        if(primer_trayecto)
                        {
                              primer_trayecto = 0;

                              a->primero = b;
                        }

                        // Fin del trayecto y decremento
                        trayectos--;

                  }

                  primer_trayecto = 1;

                  recomendaciones--;

            }


      }
}

// Funcion que extrae todos los caracteres hasta que encuentra "marca"
```

```
// La "marca" no la incluye en los caracteres extraídos.
// La busqueda comienza a partir del puntero inicio
char* extrae_carecteres_hasta_marca(char* marca, char* inicio, smsmo*
capsula_smsmo)
{
	char* caracteres;
	char* aux1;
	char* aux2;
	char buffer[TAM_PARAMETROS_RECOMENDACION];
	int i = 0;

	aux1 = inicio;
	aux2 = inicio;

	aux1 = strstr(buffer,marca)
	// aux1-->"marca"

	while((aux2) != (aux1))
	{
		buffer[i++] = (*aux2);
		aux2++;
	}
	buffer[i] = '\0';
	i = 0;

	caracteres = nuevo_mensaje(buffer,capsula_smsmo);

	return caracteres;
}


// Añade un nodo itinerario a la lista de itinerarios
void push_itinerario(itinerario** p, itinerario** f, char* id_itinerario, smsmo*
capsula_smsmo)
{
	itinerario *pc, *fc, *q;

	pc = *p;    //principio de la cola
	fc = *f;    //final de la cola

	q = nuevo_itinerario(capsula_smsmo);
	strcpy(q->identificador,id_identificador);
	q->sig=NULL;
	q->primero=NULL;
```

```
	if( fc == NULL )
		pc = fc = q;		// Si la cola esta vacia
	else
		fc = fc->sig = q; // Inserta al final de la cola

	*p = pc;
	*f = fc;
}


// Añade un nodo trayecto a la lista de trayectos
void push_trayecto(eltotrayecto** p, eltotrayecto** f, char* id_identificador,
smsmo* capsula_smsmo)
{
	eltotrayecto *pc, *fc, *q;

	pc = *p;	//principio de la cola
	fc = *f;	//final de la cola

	q = nuevo_elemento(capsula_smsmo);
	strcpy(q->identificador,id_identificador);
	q->sig=NULL;

	if( fc == NULL )
		pc = fc = q;		// Si la cola esta vacia
	else
		fc = fc->sig = q; // Inserta al final de la cola

	*p = pc;
	*f = fc;
}


// Extrae un trayecto del itinerario. Si devuelve 0, que se ha extraido el
trayecto con sus correspondientes
// parámetros Si se devuelve 1, indica que no hay trayectos en la lista.
// Ojo al extraer el trayecto lo elimina de la lista.
int pop_trayecto(eltotrayecto** p, eltotrayecto** f, char* id_trayecto, int*
destino_trayecto, time_t* h_llegada_trayecto, char* precio_trayecto, int*
modo_trayecto, char* compania_trayecto)
{
	int resultado = 1;
	eltotrayecto *pc, *fc, *q;

	pc = *p;	// principio de la cola
```

```
	fc = *f;    // final de la cola

	if( pc != NULL )  // si la cola no esta vacia
	{
		q = pc;

		// Extraemos los correspondientes parámetros
		strcpy(id_trayecto, q->identificador);
		strcpy(precio_trayecto, q->precio);
		strcpy(compania_trayecto, q->compania);
		(*destino_trayecto) = (q->destino);
		(*h_llegada_trayecto) = (q->h_llegada);
		(*modo_trayecto) = (q->modo);

		pc = pc->sig;           // actualiza la cola

		if( pc == NULL)
			fc = NULL;  // habia un solo msj

		// Eliminamos el trayecto
		free(q);

		*p = pc;
		*f = fc;
		resultado = 0;
	}

	return resultado;
}


// Funcion que elimina la lista formada por los diferentes itinerarios
// que conforman una recomendacion. Esta funcion solo se usara una vez extraídos
// todos los parámetros de los trayectos que conforman cada uno de los
itinerarios.
// Hay que pasarle un puntero al comienzo de la lista o cola como argumento
void eliminar_recomendacion_itinerarios(itinerario** p)
{
	itinerario *pc, *q;
	pc = q = *p;

	while(pc != NULL)
	{
		pc = pc->sig;
		free(q);
```

```
            q = pc;
      }
}


// Funcion que realiza un resumen de cada uno de los itinerarios dados en una
recomendacion
// Itinerario     Origen      Destino     Salida      Llegada     Precio
// Recorre cada uno de los itinerarios sin modificar su contenido.
void resumen_itinerarios(itinerario* p, char** resumen_total,smsmo*
capsula_smsmo)
{
      eltotrayecto* trayecto;       // Puntero usado para recorrer los trayectos
      struct tm *fecha;
      time_t segundos;
      char* resumen_itinerario;
      char buffer[TAM_BUFFER];
      char parametro[TAM_PARAMETROS_RECOMENDACION];
      int i = 0;  // Indice para recorrer la tabla que contiene los resumenes

      while(p != NULL)
      {
            strcpy(buffer,p->identificador);

            // Inserto un espacio en blanco
            strcat(buffer," ");

            // Veo cual es la ciudad origen
            strcat(buffer,identifica_ciudad(p->origen,capsula_smsmo));

            // Inserto un espacio en blanco
            strcat(buffer," ");

            // Veo cual es la ciudad destino
            strcat(buffer,identifica_ciudad((p->ultimo->destino),capsula_smsmo);

            // Inserto un espacio en blanco
            strcat(buffer," ");

            // Veo la fecha de salida
            segundos = p->h_salida;

            strcat(buffer,calcula_fecha(&segundos,capsula_smsmo));

            // Inserto un espacio en blanco
```

```
            strcat(buffer," ");

            // Veo la fecha de llegada
            segundos = p->ultimo->h_llegada;

            strcat(buffer,calcula_fecha(&segundos,capsula_smsmo));

            // Inserto un espacio en blanco
            strcat(buffer," ");

            sprintf(parametro,"%f",calcula_precio_itinerario(p));
            strcat(buffer,parametro);
            memset(parametro,'\0',sizeof(parametro)); // Limpia el contenido de
parametro

            // Inserto nueva linea
            strcat(buffer,"\n");

            // Ahora introduzco los detalles de cada trayecto
            trayecto = p->primero;

            while(trayecto != NULL)
            {
                  strcat(buffer,trayecto->identificador);
                  // Inserto un espacio en blanco
                  strcat(buffer," ");

                  strcat(buffer,identifica_ciudad(trayecto-
>destino,capsula_smsmo));

                  // Inserto un espacio en blanco
                  strcat(buffer," ");

                  strcat(buffer,identifica_modo(trayecto->modo,capsula_smsmo));

                  // Inserto un espacio en blanco
                  strcat(buffer," ");

                  strcat(buffer,trayecto->precio);

                  // Inserto un espacio en blanco
                  strcat(buffer," ");

                  // Veo la fecha de llegada
                  segundos = trayecto->h_llegada;
```

```
                strcat(buffer,calcula_fecha(&segundos,capsula_smsmo));

                // Inserto un espacio en blanco
                strcat(buffer," ");

                strcat(buffer,trayecto->compania);

                // Inserto nueva linea
                strcat(buffer,"\n");

                trayecto = trayecto->sig;
            }

            resumen_total[i++] = nuevo_mensaje(buffer,capsula_smsmo);
            memset(buffer,'\0',sizeof(buffer)); // Limpia el contenido del
buffer
            p = p->sig;
      }
}


// Funcion que identifica una ciudad segun su identificador y la devuelve
// en forma de cadena de caracteres
char* identifica_ciudad(int id_ciudad, smsmo* capsula_smsmo)
{
      char*
ciudades[]={"ALBACETE","ALICANTE","ALMERIA","AVILA","BADAJOZ","BARACALDO","BARCE
LONA","BILBAO","BURGOS","CACERES","CADIZ","CASTELLON","CIUDAD
REAL","CORDOBA","CUENCA","ELCHE","GERONA","GIJON","GRANADA","GUADALAJARA","HUELV
A","HUESCA","JAEN","LA CORUÃ‘A","LAS
PALMAS","LEON","LERIDA","LOGROÃ‘O","LUGO","MADRID","MALAGA","MELILLA","MURCIA","
ORENSE","OVIEDO","PAMPLONA","PALMA
MALLORCA","PALENCIA","PONTEVEDRA","SALAMANCA","SAN SEBASTIAN","SANTA CRUZ
TENERIFE","SANTANDER","SANTIAGO","SEGOVIA","SEVILLA","TARRAGONA","TERUEL","TOLED
O","VALENCIA","VALLADOLID","VITORIA","ZARAGOZA"};
      // OJO ciudad[0] = "ALBACETE" Y ciudad[52] = "ZARAGOZA"

      char* ciudad;

      ciudad = nuevo_mensaje(ciudades[id_ciudad],capsula_smsmo);

      return ciudad;

}
```

```
// Funcion que devuelve el nombre del medio de transporte segun el identificador
// 1->AVION 2->BARCO 3->BUS 4->TREN
char* identifica_modo(int id_modo, smsmo* capsula_smsmo)
{
	char* modos[]={"AVION","BARCO","BUS","TREN"};

	char* modo;

	modo = nuevo_mensaje(modos[id_modo-1],capsula_smsmo);
}


// Funcion que devuelve una cadena de caracteres que indica la fecha con el
// formato /dd/mm/aa/hh:mm
char* calcula_fecha(time_t* segundos, smsmo* capsula_smsmo)
{
	char* fecha;
	char* buffer[TAM_PARAMETROS_RECOMENDACION];
	char* parametro[TAM_PARAMETROS_RECOMENDACION];
	struct tm *fecha;

	fecha = localtime(segundos);

	// El dia
	sprintf(parametro,"/%d",fecha->tm_mday);
	strcat(buffer,parametro);
	memset(parametro,'\0',sizeof(parametro)); // Limpia el contenido de
parametro

	// El mes
	sprintf(parametro,"/%d",(fecha->tm_mon)+1);
	strcat(buffer,parametro);
	memset(parametro,'\0',sizeof(parametro)); // Limpia el contenido de
parametro

	// El año
	sprintf(parametro,"/%d",fecha->tm_year);
	strcat(buffer,parametro);
	memset(parametro,'\0',sizeof(parametro)); // Limpia el contenido de
parametro

	// La hora
	sprintf(parametro,"/%d",fecha->tm_hour);
```

```
	strcat(buffer,parametro);
	memset(parametro,'\0',sizeof(parametro)); // Limpia el contenido de
parametro

	// Los minutos
	sprintf(parametro,":%d",fecha->tm_min);
	strcat(buffer,parametro);
	memset(parametro,'\0',sizeof(parametro)); // Limpia el contenido de
parametro


	fecha = nuevo_mensaje(buffer,capsula_smsmo);
	return fecha;
}


// Funcion que calcula el precio total de un itinerario
// Devuelve el precio total calculado como un numero real
float calcula_precio_itinerario(itinerario* p)
{
	eltotrayecto* q;
	float precio_total = 0;

	if(p != NULL)
	{
		q = p->primero;

		while(q != NULL)
		{
			precio += atof(q->precio);

			q = q->sig;
		}
	}

	return precio_total;
}
```

Este código fuente es una propuesta y no debe considerarse definitivo. Por consiguiente, es necesario mejorarlo y depurarlo. No obstante, si constituye un punto de partida para implementar dicha mejora.