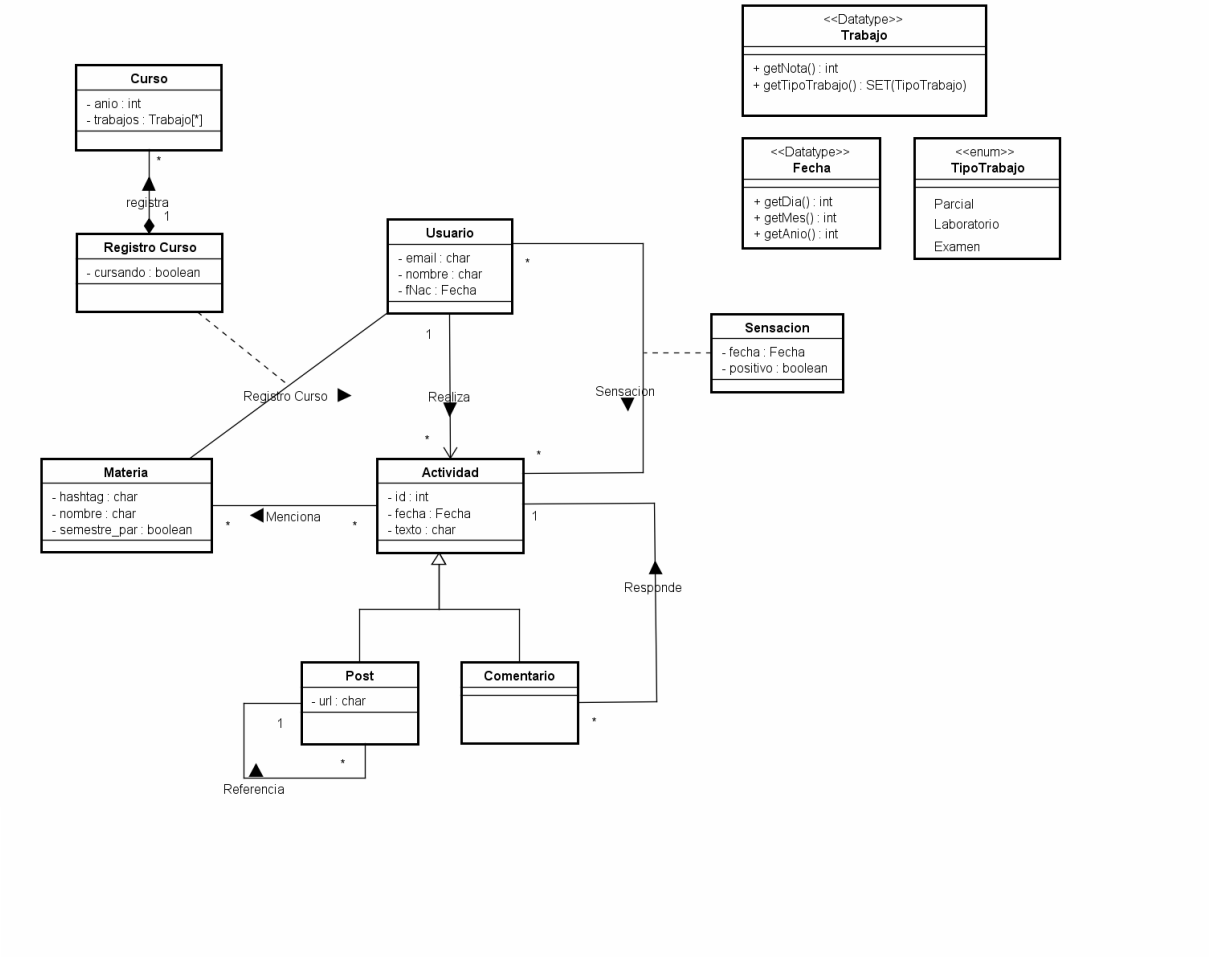
Programación 4

EXAMEN JULIO 2017 - SOLUCIÓN

Problema 1

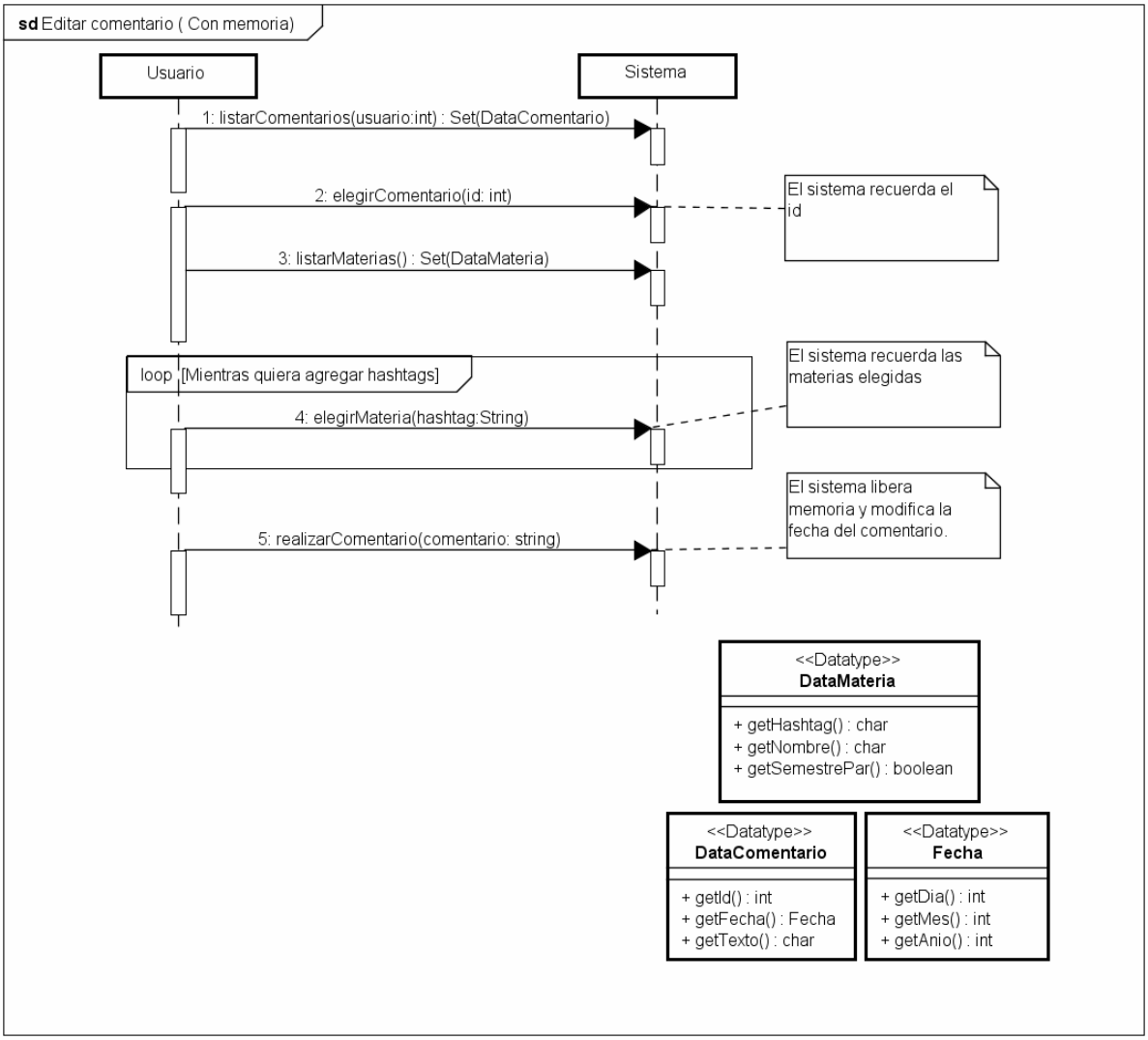
Parte a)



Restricciones:

- No existen dos usuarios con el mismo email.
- No existen dos actividades con el mismo id.
- No existen dos materias con el mismo hashtag.
- Un comentario no pueden responderse a si mismo.
- Una publicación no puede referenciarse a si misma.

Parte b)

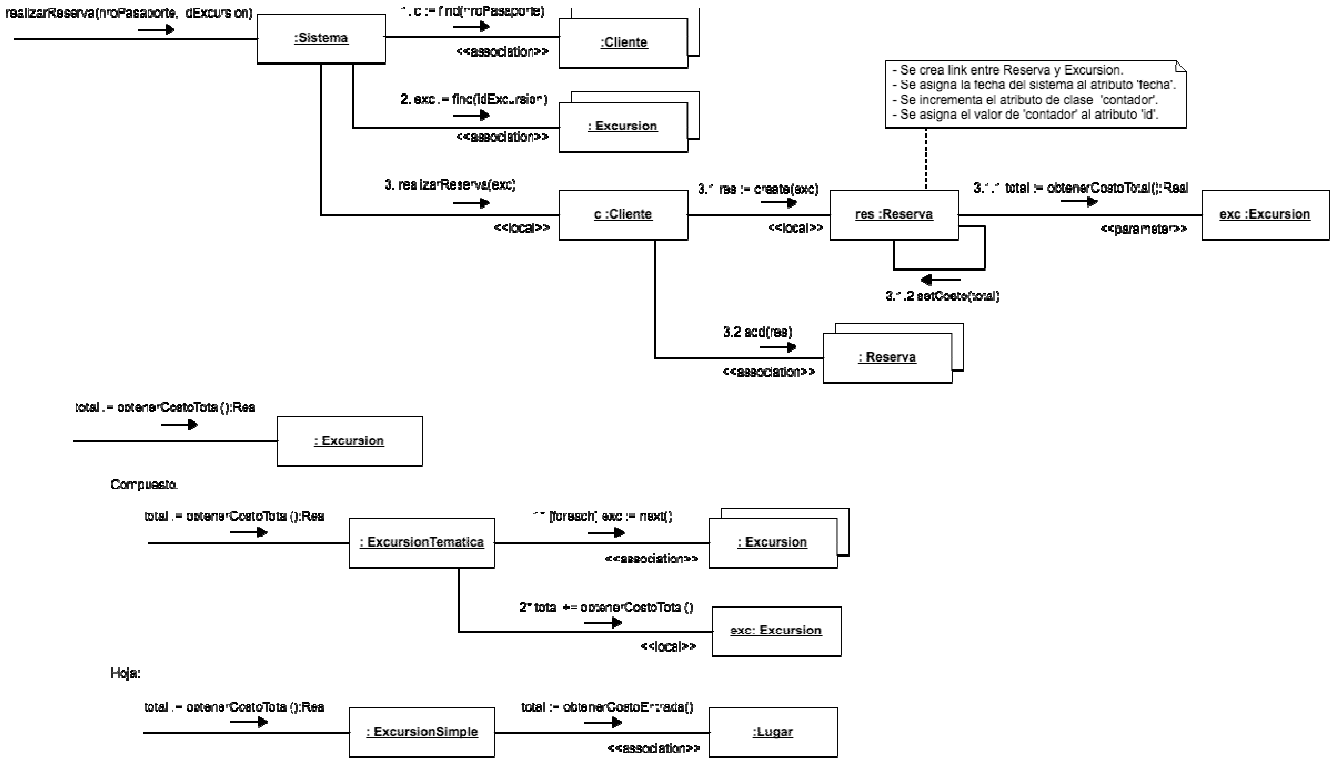

sd Editar comentario (Con memoria)
Usuario
Sistema
1: listarComentarios(usuario:int) : Set(DataComentario)
2: elegirComentario(id: int)
El sistema recuerda el id
3: listarMaterias() : Set(DataMateria)
loop [Mientras quiera agregar hashtags]
El sistema recuerda las materias elegidas
4: elegirMateria(hashtag:String)
El sistema libera memoria y modifica la fecha del comentario.
5: realizarComentario(comentario: string)
<<Datatype>>
DataMateria
+ getHashtag() : char
+ getNombre() : char
+ getSemestrePar() : boolean
<<Datatype>>
DataComentario
+ getId() : int
+ getFecha() : Fecha
+ getTexto() : char
<<Datatype>>
Fecha
+ getDia() : int
+ getMes() : int
+ getAnio() : int

Problema 2

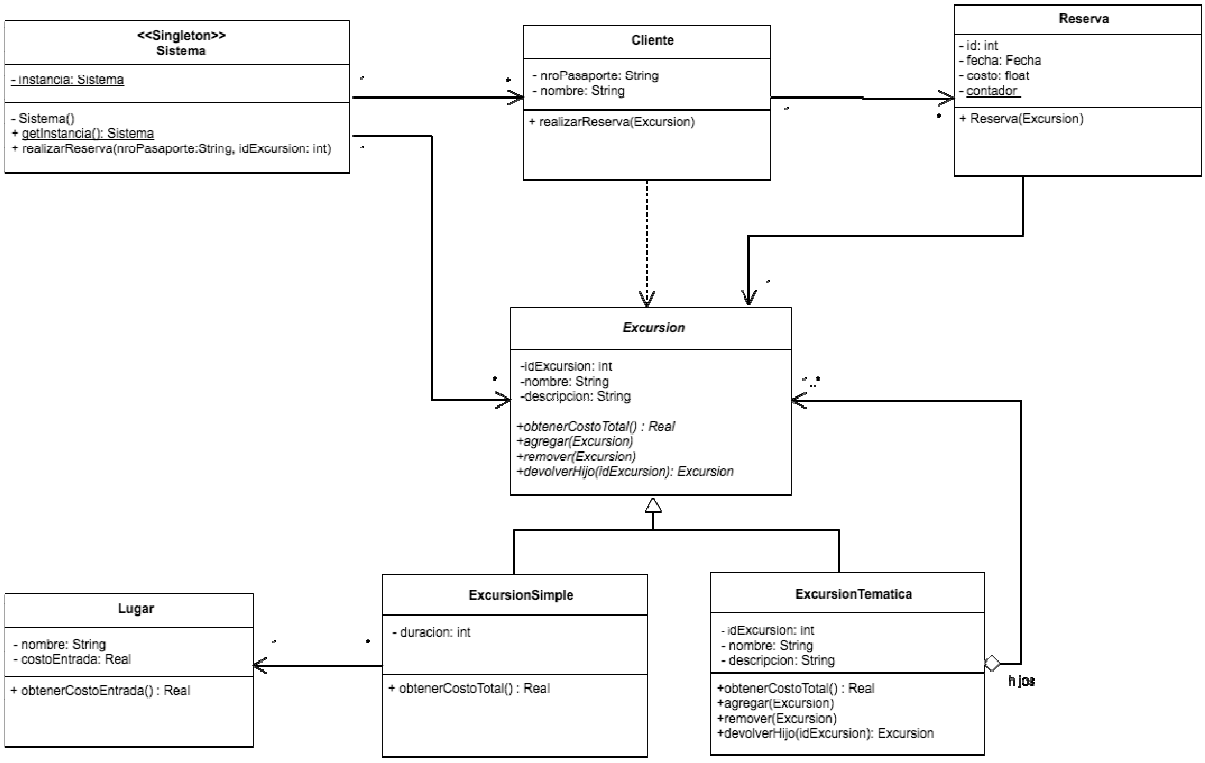
Parte I)

- Ver teórico.
- Se tiene visibilidad temporal cuando el tipo de visibilidad es local o por parámetro.

Parte II)



Se utiliza el patrón Composite para representar las excursiones. Las clases involucradas son Excursión, ExcursiónSimple y ExcursiónTematica; y sus roles son Componente, Hoja y Compuesto respectivamente.



Problema 3

```
/* DATARESULTADO.H */

class DataResultado {
private:
	std::string resultado;
public:
	DataResultado(std::string);
	std::string getResultado();
	virtual ~DataResultado();
};

/* DATARESULTADO.CPP */

DataResultado::DataResultado(std::string resultado) {
	this->resultado = resultado;
}

std::string DataResultado::getResultado() {
	return resultado;
}

DataResultado::~DataResultado() {}

/* POOLCONEXIONES.H */

class PoolConexiones {
private:
	PoolConexiones();
	static PoolConexiones* instancia;
	std::list<Conexion*> pool;
	std::map<Conexion*, bool> uso_conexiones;
public:
	static PoolConexiones* getInstancia();
	Conexion* pedirConexion();
	void liberarConexion(Conexion* con);
	virtual ~PoolConexiones();
};

/* POOLCONEXIONES.CPP */

PoolConexiones* PoolConexiones::instancia = 0;

PoolConexiones::PoolConexiones() {
	for(int i = 0; i < 5; i++) {
		Conexion* con = new Conexion();
		pool.push_front(con);
		uso_conexiones[con] = false;
	}
}

PoolConexiones* PoolConexiones::getInstancia() {
	if (instancia == 0)
		instancia = new PoolConexiones();
	return instancia;
}

Conexion* PoolConexiones::pedirConexion() {
	Conexion* con;
	if (pool.empty()) {
```

```
            con = new Conexion();
      } else {
            con = pool.front();
            pool.pop_front();
      }
      uso_conexiones[con] = true;
      return con;
}

void PoolConexiones::liberarConexion(Conexion* con) {
      if(uso_conexiones.at(con)) {
            pool.push_front(con);
            uso_conexiones[con] = false;
      }
}

PoolConexiones::~PoolConexiones() {
      std::map<Conexion*, bool>::iterator it;
      for(it = uso_conexiones.begin(); it != uso_conexiones.end();
it++)
            delete (*it).first;
}

/* CONTROLADORSERVICIOS.H */

class ControladorServicios {
private:
      ControladorServicios();
      static ControladorServicios* instancia;
public:
      DataResultado ejecutarConsulta(std::string);
      static ControladorServicios* getInstancia();
      virtual ~ControladorServicios();
};

/* CONTROLADORSERVICIOS.CPP */

ControladorServicios* ControladorServicios::instancia = 0;

ControladorServicios* ControladorServicios::getInstancia() {
      if (instancia == 0)
            instancia = new ControladorServicios();
      return instancia;
}

ControladorServicios::ControladorServicios() {}

DataResultado ControladorServicios::ejecutarConsulta(std::string
consulta) {
      PoolConexiones* pool = PoolConexiones::getInstancia();
      Conexion* con = pool->pedirConexion();
      DataResultado res = con->ejecutarConsulta(consulta);
      pool->liberarConexion(con);
      return res;
}

ControladorServicios::~ControladorServicios() {}

 /* CONEXION.H */

class Conexion {
```

```
public:
      Conexion();
      DataResultado ejecutarConsulta(std::string);
      virtual ~Conexion();
};

 /* CONEXION.CPP */

Conexion::Conexion() {}

DataResultado Conexion::ejecutarConsulta(std::string consulta) {
      return DataResultado("");
}

Conexion::~Conexion() {}
```