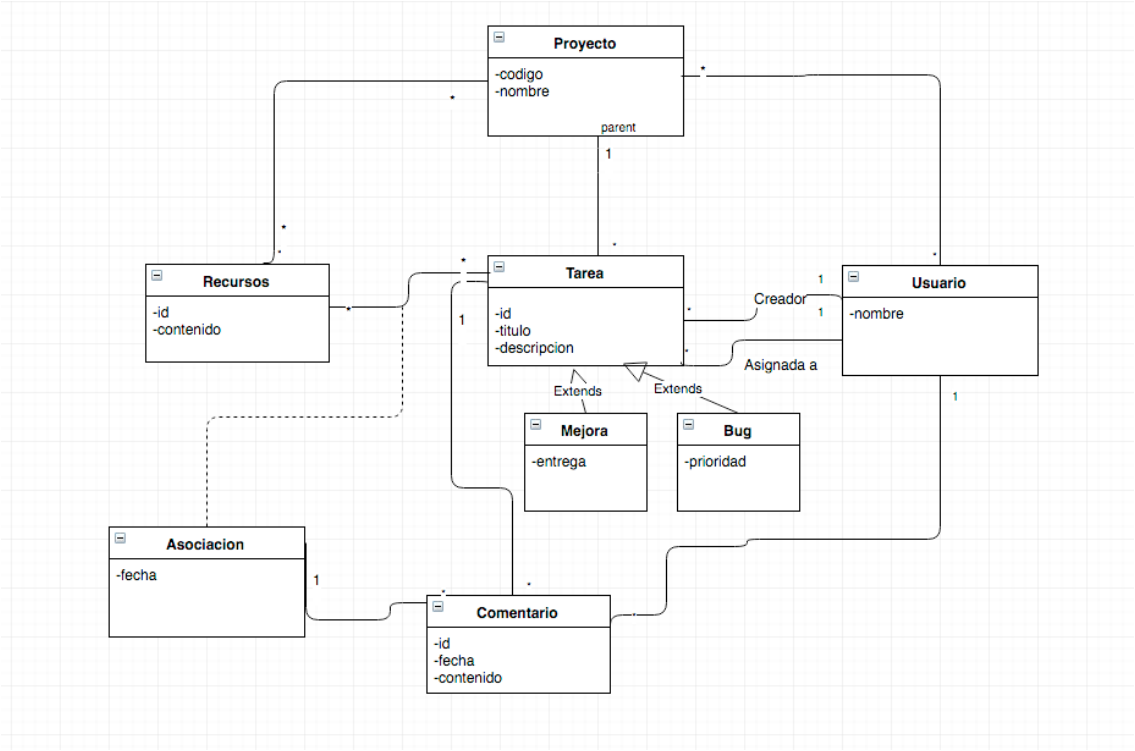
Programación 4

EXAMEN FEBRERO 2017 - SOLUCIÓN

Problema 1

(Bosquejo)

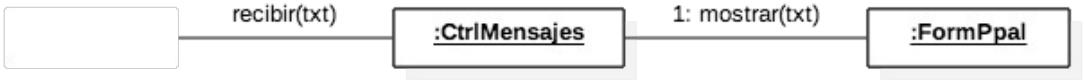


- Identificadores únicos
- Usuarios pueden comentar en tareas de proyectos a los que pertenezcan
- Usuarios pueden estar asignados a tareas de proyectos a los que pertenezcan.

Problema 2

a) Cuáles son las dos actividades (o sub-etapas) en la Etapa de Diseño?
R: Diseño de interacciones (dinámico, mediante Diag. de Comm.) y diseño de estructura (estático, mediante DCD).

b) Realice un Diagrama de Comunicación mostrando las interacciones necesarias, según lo descrito anteriormente, para recibir un mensaje.
R:



c) Qué crítica le haría a esta solución? Justifique su respuesta.
R: Que la Capa Lógica se encuentra acoplada a la Capa Presentación, habiendo una comunicación "de abajo hacia arriba", lo cual rompe los criterios definidos por la Arquitectura en Capas.

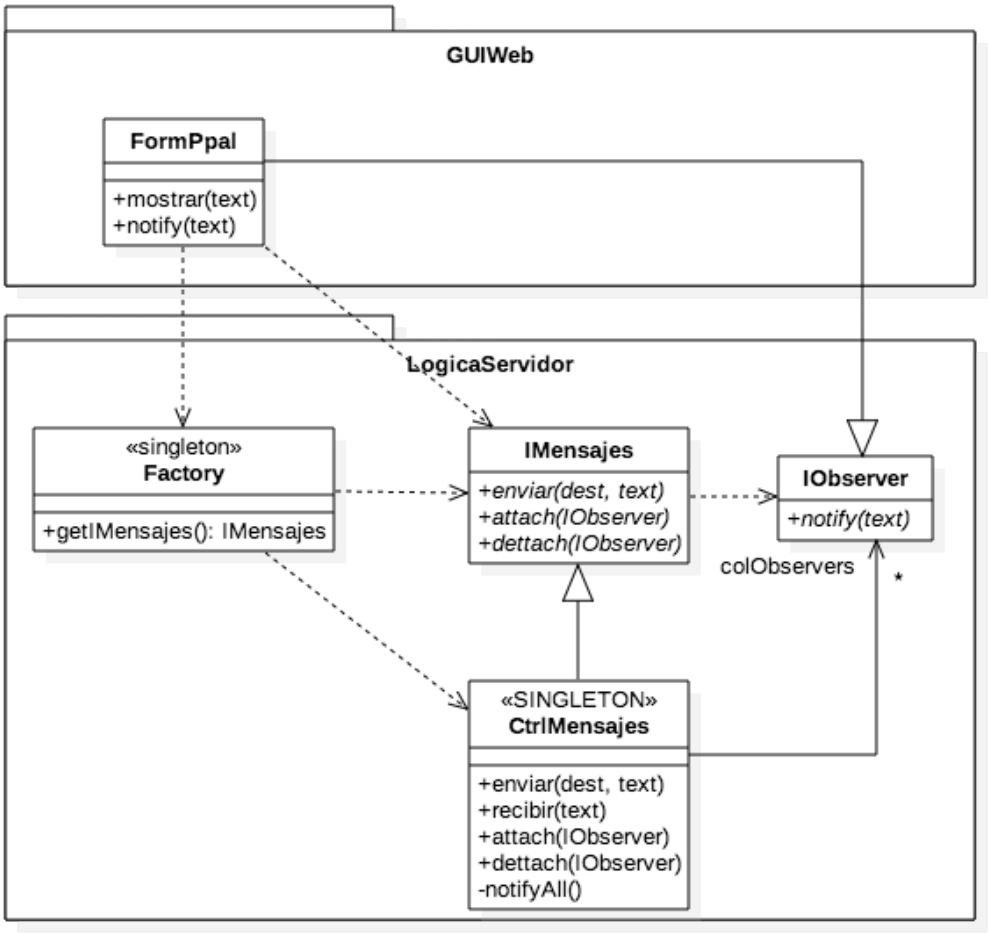
d) Qué patrón(es) de diseño propondría considerando además que la empresa se encuentra a punto de desarrollar las distintas versiones móviles (iOS, Android), cada una con su propia Capa Presentación pero utilizando la misma Capa Lógica y que se pretende que sea la Lógica la que actualice automáticamente la Presentación?
R: Aplicar el patrón Observer de la siguiente forma:

* *Subject*: el controller, que es quien recibe los mensajes y debe notificar a los diferentes formularios (Web, iOS, Android, etc.)

* *Observer*: la interfaz IObserver.

* *Concrete Observers*: los formularios que desean ser notificados de la recepción de un mensaje.

e) Realice el **nuevo** Diagrama de Clases, contemplando su solución.
R:



Problema 3

Parte 1

(a) Un objeto que permita trabajar con una instancia del procesador de comandos adquirido sin medición de duración ni log.

```
IProcesadorComando* obj = new ACMEProcesadorComando();
```

(b) Un objeto que permita trabajar con una instancia del procesador de comandos adquirido y medición de duración.

```
IProcesadorComando* obj =
            new MedidorTiempo(new ACMEProcesadorComando());
```

(c) Un objeto que permita trabajar con una instancia del procesador de comandos adquirido y con ambas funcionalidades: medición de duración y log.

```
IProcesadorComando* obj =
            new Logger(new MedidorTiempo(new ACMEProcesadorComando()));
```

Tambien se considera correcta la solucion:

```
IProcesadorComando* obj =
            new MedidorTiempo(new Logger(new ACMEProcesadorComando()));
```

Parte 2

```
/****************** IProcesadorComando *********************/

// IProcesadorComando.h

// Interfaz IProcesadorComando
class IProcesadorComando {
public:
    virtual Comando recibir(Pedido& pedido) = 0;
    virtual Respuesta ejecutar(Comando& cmd) = 0;
    virtual void enviar(Respuesta& respuesta) = 0;
};


/****************** Interceptor *********************/

// Interceptor.h

// Clase abstracta Interceptor
class Interceptor : public IProcesadorComando {
public:
    Interceptor(IProcesadorComando* procesadorComando);
    virtual Comando recibir(Pedido& pedido);
    virtual Respuesta ejecutar(Comando& cmd);
    virtual void enviar(Respuesta& respuesta);
    virtual ~Interceptor() = 0;
private:
    IProcesadorComando* procesadorComando;

};

// Interceptor.cpp

Interceptor::Interceptor(IProcesadorComando* procesadorComando) {
    this->procesadorComando = procesadorComando;
}

Comando Interceptor::recibir(Pedido& pedido) {
    return this->procesadorComando->recibir(pedido);
}

Respuesta Interceptor::ejecutar(Comando& cmd) {
    return this->procesadorComando->ejecutar(cmd);
}
```

```
void Interceptor::enviar(Respuesta& respuesta) {
    return this->procesadorComando->enviar(respuesta);
}


/****************** MedidorTiempo *********************/

// MedidorTiempo.h

// Clase que agrega funcionalidad de medir duracion de un procesador de
comando
class MedidorTiempo : public Interceptor {
public:
    MedidorTiempo(IProcesadorComando* procesadorComando);

    Comando recibir(Pedido& pedido);
    void enviar(Respuesta& respuesta);

    virtual ~MedidorTiempo();
private:
    Cronometro* crono;
};

// MedidorTiempo.cpp

MedidorTiempo::MedidorTiempo(IProcesadorComando* procesadorComando) :
                                                    Interceptor(procesadorComando) {
    crono = new Cronometro();
}

Comando MedidorTiempo::recibir(Pedido& pedido) {
    crono->prender();
    return Interceptor::recibir(pedido);
}

void MedidorTiempo::enviar(Respuesta& respuesta) {
    Interceptor::enviar(respuesta);
    crono->apagar();
    Time d = crono->darDuracion();
    // hace algo con la duracion d
}


/****************** Logger *********************/

// Logger.h

// Clase que agrega la funcionalidad de registrar un log de las ejecuciones de

// un procesador de comando
class Logger : public Interceptor {
public:
    Logger(IProcesadorComando*);

    Respuesta ejecutar(Comando&);
    void eliminarRango(FechaHora&, FechaHora&);

    virtual ~Logger();
private:
    vector<ItemLog*> items;

};

// Logger.cpp

Logger::Logger(IProcesadorComando* pc) : Interceptor(pc) {
}

Respuesta Logger::ejecutar(Comando& cmd) {
    Respuesta res = Interceptor::ejecutar(cmd);
    items.push_back(new ItemLog(cmd.getFechaHora(), cmd.getName()));
```

```
    return res;
}

void Logger::eliminarRango(FechaHora& fechaDesde, FechaHora& fechaHasta) {
    ItemLog* actual;
    vector<ItemLog*>::iterator it;
    for(it = items.begin(); it != items.end(); ) {
        actual = *it;
        if (actual->darFechaHora() >= fechaDesde &&
                    actual->darFechaHora() <= fechaHasta) {
            it = items.erase(it);
            delete(actual);
        } else {
            ++it;
        }
    }
}

Logger::~Logger() {
}


/****************** ItemLog *********************/

// ItemLog.h

class ItemLog {
public:
    ItemLog(FechaHora&, string);

    FechaHora darFechaHora();

    ~ItemLog();
private:
    FechaHora fechaHora;
    string nombreCmd;
};

// ItemLog.cpp

ItemLog::ItemLog(FechaHora& fh, string nom) {
    fechaHora = fh;
    nombreCmd = nom;
}

FechaHora ItemLog::darFechaHora(const ItemLog& orig) {
    return fechaHora;
}

ItemLog::~ItemLog() {
}
```