Guia de implementación Qapital

ULTIMA ACTUALIZACION: 15/01/2005

Si quieres empezar a trabajar en qapital, por favor lea los siguientes requisitos:

- Los nombres de las clases empiezan con mayúscula y llevan las tres primeras consonantes de la capa a la que pertenezcan, el archivo se debe llamar igual a la clase pero en minúsculas.
  Ejemplo:
  GCLPrincipal: GUI Cliente

- Los nombres de las funciones empiezan con minúsculas, no se separan con la raya abajo sino con la letra en mayúscula. Ejemplo
  nombreDeVariable

- Los nombres de variables, clases, etc deben ser lo mas claros posibles, pensando en que cualquier otro programador puede leer el código y necesita entenderlo.

- Las variables temporales llevan el prefijo tmp, Ejemplo:
  tmpVariable

- Las llaves de bloques de código van una debajo de otra y alineadas.

```
Funcion()
{
    // Codigo
}
```

- En el constructor y destructor de una clase que herede de un objeto Qt, por favor coloque una linea como la siguiente:

```
Clase::Clase()
{
        INITQPC
        ...
}
```

  Para el destructor se cambia **INITQPC** por **FINQPC**.
  Si no hereda de Qt por favor añada: qDebug("[Construyendo *NOMBREDECLASE*]"); en el constructor y en destructor cambie Construyendo por Destruyendo, esto hace mas fácil la localización de violaciones de segmento.

- Estudie detenidamente todos los módulos del Sistema Base para (re)utilizarlos en el proyecto.

- Si desarrollas alguna clase, modulo para algún Sistema Base asegurate de incluir el archivo de cabecera en el archivo de cabecera del Sistema Base. Ejemplo:
  Suponiendo creo la clase SbXmlPAQUETE, debo incluir el archivo sbxmlpaquete.h en sbxml.h

- Avise al director del proyecto o a un director de sub-proyecto según sea el caso, en que archivos va a trabajar, y que modificaciones o inclusiones piensa hacer. Si es subdirector avise los cambios al director.

- Las cadenas deben ser construidas con tr() para que puedan ser traducidas por el equipo de traducción, ejemplo:
  tr("Esto es una cadena").

- Si considera que algo hace falta en su código, agregue un comentario con la palabra TODO: y un mensaje explicativo de lo que hace falta hacer, de esta forma podemos hacer un grep para mirar que hace falta.
  Ejemplo:
  // TODO: Hacer interfaz gráfica para generar configuración del cliente.

- Si considera que algo no se hace bien o falla en su código, agregue un comentario con la palabra FIXME: y un mensaje explicativo de lo que hace falta hacer, de esta forma podemos hacer un grep para mirar que hace falta.
  Ejemplo:
  // FIXME: Hacer interfaz gráfica para generar configuración del cliente.

- Se debe comentar todo el código que se escriba siguiendo los parámetros de Doxygen, el cual es el sistema de documentación que hemos elegido para el proyecto, por ejemplo al iniciar las funciones debe escribir /** Descripción de la función */, si tienes parámetros utilice el comando @param de doxygen, si hay referencia a otras funciones o clases, utilice el comando @see, antes de comenzar las clase utilice el comando @short para dar un descripción, y abajo la descripción completa, para mas detalles consulte la documentación de doxygen.
- Lleve un registro diario con fecha de las modificación que hace a su código, al momento de enviar el código anexe ese archivo, la sintaxis de ese archivo es de la forma: (lea Changelog)

    - DIA/MES/AÑO: Texto del cambio. (su@correo.org)

    Ejemplo:

    - 11/08/2004: Se comenzo el Changelog. (krawek@linuxmail.org)

- Lleve un registro de las cosas que desea que se hagan en un futuro (lea TODO LIST).
- Cuando termine de escribir su código, debe pasar en un directorio comprimido el código de los archivos que ha modificado, un archivo Changelog y un archivo TODO.

Recuerde que si no sigue los pasos anteriores su código no sera incluido en el proyecto.

--El director del proyecto.