PATRONES DE SOFTWARE, TEMARIO

Desde las horas extras

https://desdelashorasextras.blogspot.com/

Realizado por
José Luis Bautista Martín

Julio de 2017

Esta hoja está en blanco para facilitar la impresión de este documento a doble página.

ACERCA DEL INSTRUCTOR

José Luis Bautista Martín, Ingeniero de Sistemas, con maestría en "investigación en ingeniería de Software".

Mi experiencia laboral en cuanto a desarrollo de software abarca desde tecnologías "legacy", hasta tecnologías de vanguardia, poniendo siempre una especial atención en la construcción de software escalable, modular y sostenible.

Igualmente estoy especializado en la interconexión de diversos sistemas y plataformas para conseguir una solución coherente entre la tecnología actual en producción y nuevas tecnologías del mercado.

Una de mis inquietudes actuales es simplificar el desarrollo de software, permitiendo mediante herramientas generadoras de código, patrones de software, programación orientada a aspectos o simplemente interfaces sencillas y claras que el programador se concentre en resolver los problemas propios de la solución a implementar (esto es, los requisitos de negocio a representar en forma de software) y no se tenga que preocupar de tareas repetitivas, generalidades de los sistemas, o problemas técnicos, que no hacen más que distraerle de sus verdaderos objetivos.

ACERCA DE ESTE DOCUMENTO

La misión de este documento es exponer los objetivos, mecánica y temerarios planteados para el curso "Patrones de Software".

OBJETIVOS DEL CURSO

Los patrones de software son soluciones previamente establecidas (y probadas como optimas) a problemas conocidos y repetitivos dentro del desarrollo de software.

El curso de "Patrones de Software" pretende proporcionar herramientas para crear un escenario en el que se favorezca la creación de software de calidad, escalable y funcional.

A la vez que se revisan conceptos básicos para la ingeniera de software (para establecer un contexto de inicio) se estudiaran una selección de los más útiles patrones de software. Por otro lado y a modo de complemento se revisar una colección de los "peores" patrones de software, o anti-patrones (comportamiento y metodologías perjudiciales para la construcción de sistemas), de forma que sirva como elementos comparativo.

.

CONTENIDO

METODOLOGÍA

Se distribuirá el curso en tres sesiones de tres horas cada una en las cuales, después de una introducción de los temas a plantear, se realizan talleres prácticos de los patrones de software, los cuales serán realizados en equipos de a dos (dos personas compartiendo una computadora).

REQUISITOS

Es necesario una laptop por cada dos personas, y tener conocimientos promedios de programación en C#. La computadora debe tener instalador Visual Studio 2010 o superior.

- Visual Studio 2015 (o superior).
- SQL Express 2008 o superior con el cliente (Debe haber una instancia en el cliente).
    - https://www.microsoft.com/es-mx/download/details.aspx?id=1842
- System.Data.SQLite (buscar en la pagina Setups for 32-bit Windows (.NET Framework 4.6)).
    - https://system.data.sqlite.org/index.html/doc/trunk/www/downloads.wiki
- Vamos a realizar un taller de Android Studio (solo uno), con lo que si lo desean también pueden tenerlo instalado, en dicho caso les recomiendo también un celular y un cable USB.
    - https://developer.android.com/studio/index.html?hl=es-419
- Les proporcionare igualmente un script de Ruby para facilitar la configuración a base de datos, si desean usarlo, deben instalarse Ruby, si no deberán hacer la configuración manualmente.
    - https://rubyinstaller.org/downloads/

- DB Browser for SQLite.
    - http://sqlitebrowser.org/

TEMARIO

1. **Presentación de objetivos**

2. **El desarrollo de software en la empresa**

En este apartado se trataran temas propios de la ingeniera de software y el desarrollo de software en la empresa con intención de establecer un contexto previo

a) Acerca de la ingeniería de software

Ref: http://desdelashorasextras.blogspot.mx/2017/04/diferencias-entre-ciencias-la.html

b) Escenarios dentro del desarrollo de software

Ref: http://desdelashorasextras.blogspot.mx/2016/08/capicuagen-desarrollo-de-software-en-la.html

c) Construcción de una fábrica de software

Ref: https://desdelashorasextras.blogspot.mx/2016/10/diseno-de-una-fabrica-de-software.html

3. **Acerca de los programación orientada a objetos**

Se analizara los principios básicos de la programación orientada a objetos, para establecer los fundamentos solos los que se sustentan los patrones de software

a) Principios generales de la orientación a objetos

Ref: https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

b) **Principios SOLID**

Ref: https://es.wikipedia.org/wiki/SOLID

4. Introducción a los patrones de software

Ref: https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o

Breve introducción a los patrones de software, su origen y su utilidad.

c) **Breve historia de los patrones de software**

d) **Tipos de patrones de software**

e) **Anti patrones de software**

5. Explicación de patrones de software

Se explicaran y se ejemplificaran, los siguientes patrones de software:

- Factory
- Abstract Factory
- Adapter
- Builder
- Chain of Resposibility
- Command
- Composite
- Decorator
- Fascade
- Model View Controller
- Observer
- Proxy
- Strategy
- Visitor

6. Hediondez del código

Ref: https://es.wikipedia.org/wiki/Hediondez_del_c%C3%B3digo

En este apartado se trata la "hediondez del código", un concepto por el cual un software que aparentemente funciona bien, oculta graves problemas en su interior que pueden emerger en cualquier momento. Se revisaran los siguientes conceptos

- Código duplicado.
- Clase grande.
- Demasiados parámetros.
- Envidia de características.
- Herencia rechazada.
- Complejidad artificiosa.

7. Anti-patrones de software

Los anti-patrones de software son la mejor forma de hacer algo mal. Aquí se estudiaran con intención de evitarlos.

Ref: https://es.wikipedia.org/wiki/Antipatr%C3%B3n_de_dise%C3%B1o

Los anti-patrones para estudiar a:

- Base de datos como comunicador de procesos.
- Clase Gorda.
- Re-dependencia.
- Acoplamiento secuencial.
- Modelo de dominio anémico.
- YAL (Yet Another Layer, y otra capa más).
- Ancla del barco.
- Código espagueti.
- Martillo de oro.
- Reinventar la rueda.
- No inventado aquí.
- Otra reunión más lo resolverá.
- Proyecto del día de la marmota.
- Si funciona, no lo toques.

8. **Conclusiones**