Optimización del Cálculo Eléctrico y Generación de Planos preliminares para Generadores Hidráulicos son aplicaciones desarrolladas con total exclusividad para la línea Hydro de INDAR Electric S.L.

El Cálculo Eléctrico es una herramienta capaz de lograr el generador hidráulico óptimo del catálogo de ofertas de la empresa INDAR.

El Generador de planos preliminares permite anexar planos a la fase de oferta de generadores hidráulicos.

ÍNDICE

1. INTRODUCCIÓN

1.1 Antecedentes

En la actualidad, la empresa dispone de una aplicación llamada *El Configurador de Ofertas*, mediante la cual se reduce el tiempo de respuesta y el gasto de recursos humanos a la hora de afrontar una petición de oferta a la unidad de trabajo Hydro. La resolución de la oferta consistirá en la realización de una forma automática de los cálculos eléctricos y mecánicos, cálculo de los costes, selección y anexión de la documentación necesaria, especificación de los puntos de inspección, creación del documento final de oferta y finalmente, volcado de información en la Base de datos de Ofertas existente en la actualidad. (Para más información sobre *El Configurador de Ofertas* consultar el Anexo I)

Este proyecto consta de: el Subproyecto 1 o Cálculo Eléctrico y Subproyecto 2 o Generador de Planos.

La labor en el Subproyecto 1 o Cálculo Eléctrico será un trabajo de reingeniería. Una vez separado el módulo del Cálculo Eléctrico existente en El Configurador de Ofertas actual, se procederá a analizar con detalle el código, extrayendo, fórmulas, algoritmo o pasos a seguir para realizar el cálculo eléctrico de una máquina. Una vez obtenido el algoritmo o proceso a seguir, se presentará un documento al grupo de Técnica eléctrica con el fin de depurar los fallos existentes o especificar las mejoras que se desean añadir. Una vez detallado y documentado se procederá a implementar el nuevo Cálculo Eléctrico, cálculo que será integrado en el actual Configurador de Ofertas.

El Subproyecto 2 o Generador de Planos, es un nuevo módulo que se utilizará integrado en el actual Configurador de Ofertas o como módulo independiente. Será una herramienta ágil y fácil de utilizar que anexará un plano del generador ofertado, un plus de calidad para el actual Configurador de Ofertas.

1.2 Presentación de la empresa INDAR

Historia:

1940: Los hermanos Larrañaga y Ormazabal fundan INDAR.
1965: INDAR recibe el título de “Empresa Ejemplar”.
1997: INDAR se incorpora al GRUPO INGETEAM.
2000: Se inaugura la nueva planta en Beasain.
2003: Las distintas unidades de Negocio de INDAR experimentan un crecimiento considerable.
2006: INDAR se ha convertido en uno de los líderes mundiales en la fabricación de generadores hidráulicos, motores sumergibles de alta potencia y en el sector eólico para el que se han suministrado generadores que suponen mas del 12% de la potencia instalada en el mundo.

Descripción de INDAR:

Entre las empresas del grupo INGETEAM, INDAR con un equipo humano compuesto por mas de 750 trabajadores, destaca por su trayectoria, que viene avalada por la construcción de miles de motores y generadores para los sectores Industrial, Energético y Naval. Sus máquinas son conocidas y apreciadas por su robustez, calidad y máxima fiabilidad.

Además, INDAR Electric, S.L es una empresa que invierte de manera continuada en I+D+i, cuyo departamento está formado por más de 40 ingenieros y licenciados altamente cualificados y dotado con los medios más avanzados para el diseño y desarrollo de máquinas con tecnología propia, garantizando la máxima flexibilidad y prestaciones y en estrecha colaboración con sus clientes.

INDAR cuenta con una planta en Beasain y otra en Idiazabal, y posee un campo de actividad bastante amplio: abarca desde la energía (eólica, hidroeléctrica, cogeneración y térmicas), la industria (sidurergia y metalurgia), el campo naval (propulsión, accionamientos varios, generación eléctrica, motores sumergibles) y ferroviario (Tracción, generación) hasta las infraestructuras (instalaciones de bombeo, planta desaladoras).

Productos y Servicios:

En INDAR fabricamos máquinas eléctricas rotativas con la máxima garantía de robustez y calidad y en estrecha colaboración con nuestros clientes. Nuestras nuevas instalaciones, inauguradas en Octubre del año 2000, están dotadas de los medios técnicos mas avanzados para el diseño y desarrollo de máquinas con la máxima garantía y fiabilidad.

La amplia gama de productos de INDAR está dedicada a satisfacer las más variadas necesidades en los sectores Industrial, Energético, Naval, Ferroviario, de Infraestructuras y Medioambiente.

Entre estos productos encontramos la siguiente gama de productos:

- **COGERACIÓN:**

  Generadores síncronos para turbinas de vapor, gas y motores diesel desde 1.250 kVA hasta 35.000 kVA y tensión desde 690 V hasta 15 kV. Para todas las máquinas aislamiento H, IP-23 hasta IP-56 y horizontales.

- **HIDRÁULICA:**

  Generadores sícronos desde 1.250 kVA hasta 40.000 kVA y tensión desde 690 V hasta 15 kV. Para todas las máquina aislamiento H, IP-23 hasta IP-56, verticales y horizontales.

- **GENERADORES EÓLICOS:**

  Generadores asíncronos desde 850 kW hasta 6.000 kW y tensión desde 690 V hasta 15 kV. Para todas las máquinas aislamiento H, IP-23 hasta IP-56 y horizontales.

- **MOTORES ASINCRONOS:**

  Motores de jaula de ardilla y anillos rozantes desde 400 kW hasta 15.000 kW y tensión desde 690 V hasta 15 kV. Para todas las máquinas aislamiento H, IP-23 hasta IP-56, verticales y horizontales.

- **MOTORES SINCRONOS:**

  Motores desde 1.000 kW hasta 15.000 kW y tensión desde 690 V hasta 15.000 V. para todas las máquinas aislamiento H, IP-23 hasta IP-56 y horizontales.

- **MOTOTES CC.:**

  Motores que llegan hasta 4.000 kW

- **MOTORES SUMERGÍBLES:**

  Son motores desde 1.000 kW hasta 10.000 kW, con una tensión desde 690 V hasta 15 kV y se sumergen hasta 1.000 m de profundidad en el mar.

Además INDAR Electric, S.L, cuenta con una Unidad de Negocio, situada en Idiazabal, y destinada a la reparación de todas las máquina que se encuentran fuera del periodo de garantía.

2. DOCUMENTO DE OBJETIVOS DE PROYECTO

2.1 Descripción:

Es un proyecto a desarrollar por Itziar Uranga que se corresponde con el Proyecto Fin de carrera de la titulación ITIS (Ingeniería Técnica en Informática de Sistemas) en el curso académico 2009/2010.

Se desarrollará en y para la empresa Indar Electric. Tendrá una duración aproximada de 8 meses. Será supervisado desde Indar por Gorka Domínguez, desarrollador de software de la sección – HYDRO Generators, y por el profesor Germán Rigau de la universidad del País Vasco

El proyecto surge por la necesidad del departamento Comercial de la unidad de negocio de generadores hidráulicos HYDRO, de una herramienta capaz de conseguir en el momento, cálculos costosos que lograrán obtener el generador hidráulico que mas se le ajuste al cliente. Además introduciremos una opción muy ambiciosa desde el punto de vista de la venta, la posibilidad de generar los planos en el momento, de la máquina que se oferta al cliente, con todos sus componentes.

La aplicación se integrará en la Intranet y Extranet de la empresa Indar Electric, para que todo usuario con acceso puedan utilizarla.

Se trata por tanto de la creación de módulos ejecutables para la automatización y optimización de cálculo eléctrico de generadores hidráulicos y generación de planos preliminares,

2.2 Objetivos:

Por un lado, se creará un ejecutable capaz de efectuar el proceso de cálculo eléctrico para la obtención del generador hidráulico óptimo del catálogo de ofertas Indar, según ciertas características de entrada por parte del cliente. El ejecutable, creará un archivo en el que quedará reflejado todo el proceso y cálculos realizados por la aplicación para lograr dicho resultado.

Por otro lado, se desarrollará otro ejecutable capaz de manejar información contenida en un fichero generado de forma automática por el Configurador de ofertas (consultar Anexo I para comprender la labor del Configurador de ofertas) o bien creado a través de un formulario por el usuario. Mediante el uso de una librería de imágenes, generará planos de oferta de generadores hidráulicos Indar, con cotas y textos.

Se diseñará todo de forma que preserve la privacidad de la información de Indar, tales como: el catálogo de ofertas de generadores hidráulicos, el proceso y cálculos para obtener una máquina óptima, los ficheros de configuración eléctrica base que hay por cada tipo de máquina, etc.

2.3 Método de trabajo:

El proyecto se aborda con la intención de ser presentado en el mes de junio de 2010. Para lograr el objetivo se imponen un mínimo de 35 horas semanales a llevar a acabo en días laborales en la empresa Indar.

Al ser un proyecto real, la elección de tecnologías para su desarrollo, está sujeto a las preferencias de la empresa y a la propiedad de licencias por parte de Indar.

Se dividirá el trabajo en distintas iteraciones, siguiendo el Proceso Unificado de Desarrollo. Marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso. Las iteraciones de cada subproyecto, siguen el desarrollo en cascada, desarrollo que ordena rigurosamente las etapas del ciclo de vida del software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

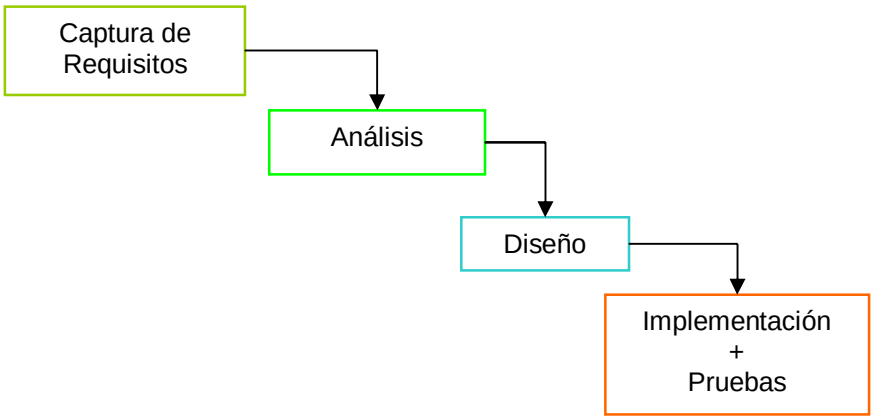


Imagen 1 – Desarrollo en cascada

- *Captura de requisitos:* Debido a que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema.

- *Análisis:* El ingeniero de software debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.

- *Diseño:* El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación. Presenta la estructura de los datos, la arquitectura del software y la caracterización de la interfaz

- *Implementación:* El diseño debe traducirse en código descifrable para la maquina. El paso de implementación realiza esta tarea. Si el diseño se realiza de una manera detallada la implementación puede realizarse de una forma más sencilla.

- *Pruebas:* Una vez que se ha generado el código comienza la fase de pruebas. La prueba se centra en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

Disponemos de un disco duro en red del que se hacen backups diarios, en se almacena una copia de los documentos generados tanto para la memoria como para la empresa. Además se guarda dicha información en CD´S y en diferentes equipos.

Se determinará junto con mi supervisor y el director de proyectos (los dos de la empresa Indar) que información es confidencial y que información no lo es para la posterior presentación (exposición) del proyecto.

Se acuerdan reuniones puntuales con el tutor de proyecto de la UPV, reuniones semanales con mi supervisor y el director de proyectos INDAR, para la exposición de avances, problemas, dudas… (Para todo lo que pueda surgir). Tras cada reunión se redactará un acta detallando las tareas a realizar por cada miembro, para la siguiente reunión.

Se llevará un plan de trabajo diario anotando todas las tareas llevadas a cabo a lo largo del día.

2.4 Alcance:

2.4.1 Esquema de descomposición de trabajos:

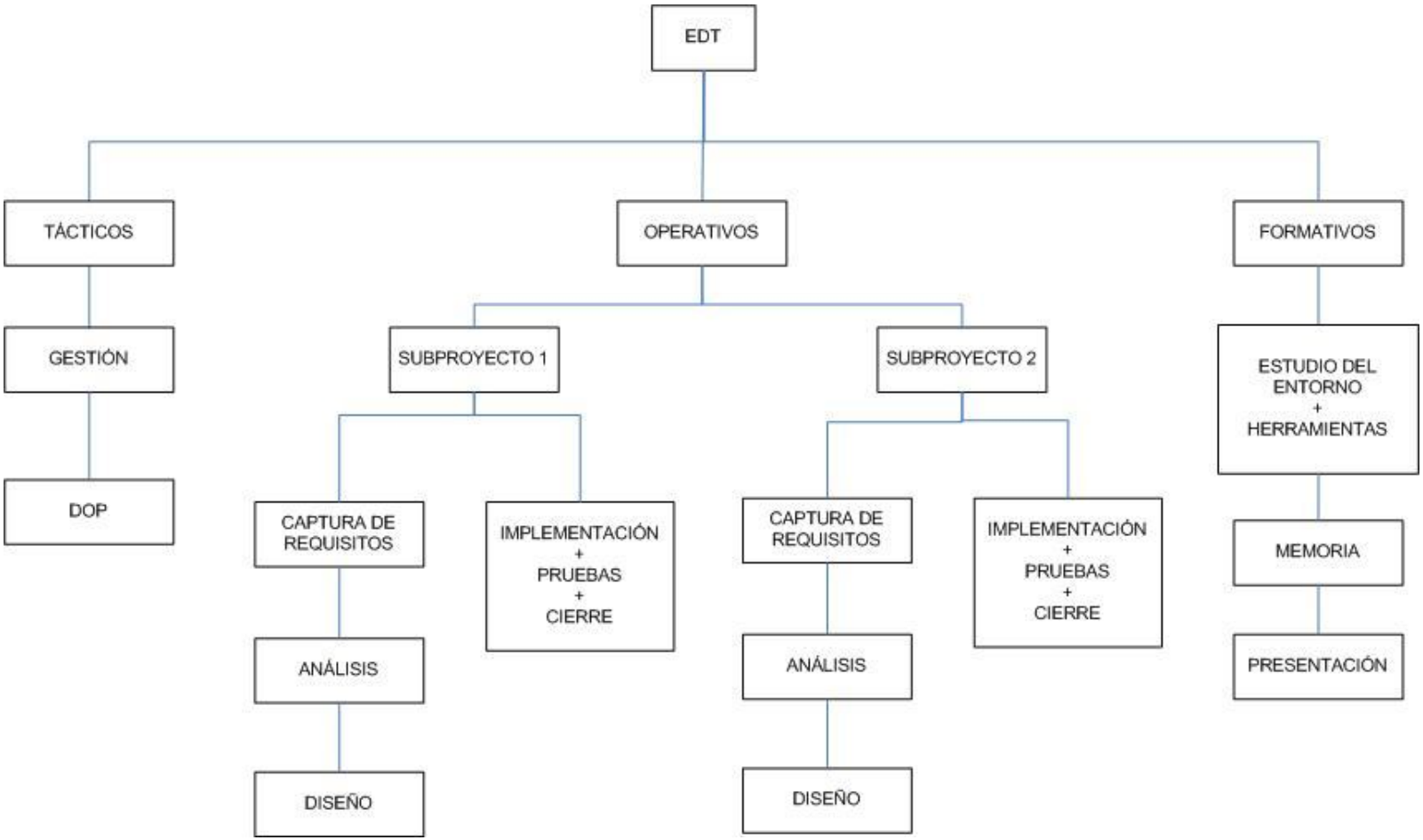


Imagen 2 - EDT

2.4.2 Estimación de esfuerzo:

Se destinarán en torno a 1000h/persona para el desarrollo del proyecto.

▪ Fase 1:

Estimación de esfuerzo: 2 días * 7 h/día

Tareas:

DOP:

- Descripción
- Objetivos
- Método de trabajo
- Alcance
- Planificación temporal
- Plan de contingencia
- Factibilidad
- Reuniones

Gestión de proyecto

Tipo: informe

▪ Fase 2:

Estimación de esfuerzo: Dos semanas * 7 h/día

Tareas:

Estudio del entorno y de las herramientas
Gestión de proyecto

▪ Fase 3:

Estimación de esfuerzo: 3 días * 7h/día

Captura de requisitos subproyecto 1

- Roles
- Casos de uso
- Modelo de dominio
- Diseño de las pantallas
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 4:

Estimación de esfuerzo: Dos semanas * 7h/día

Análisis subproyecto 1

- Diagramas de secuencia del sistema
- Contratos
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 5:

Estimación de esfuerzo: Tres semanas * 7h/día

Diseño subproyecto 1

- Diagramas de iteración
- Diagramas de clase
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 6:

Estimación de esfuerzo: Seis semanas * 7h/día

Tareas:

- Implementación subproyecto 1
- Pruebas subproyecto 1
    - Cierre subproyecto 1
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 7:

Estimación de esfuerzo: Una semana * 7h/día

Captura de requisitos subproyecto 2

- Roles
- Casos de uso
- Modelo de dominio
    - Diseño de las pantallas
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 8:

Estimación de esfuerzo: Tres semanas * 7h/día

Análisis subproyecto 2

- Diagramas de secuencia del sistema
- Contratos
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 9:

Estimación de esfuerzo: Un mes * 7h/día

Diseño subproyecto 2

- Diagramas de iteración
- Diagramas de clase
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 10:

Estimación de esfuerzo: Seis semanas * 7h/día

Tareas:

- Implementación subproyecto 2
- Pruebas subproyecto 2
- Cierre subproyecto 2
- Reuniones
- Gestión de proyecto

Tipo: Informe

▪ Fase 11:

Estimación de esfuerzo: Dos semanas * 7h/día

Tareas:

- Memoria

Tipo: Informe

▪ Fase 12:

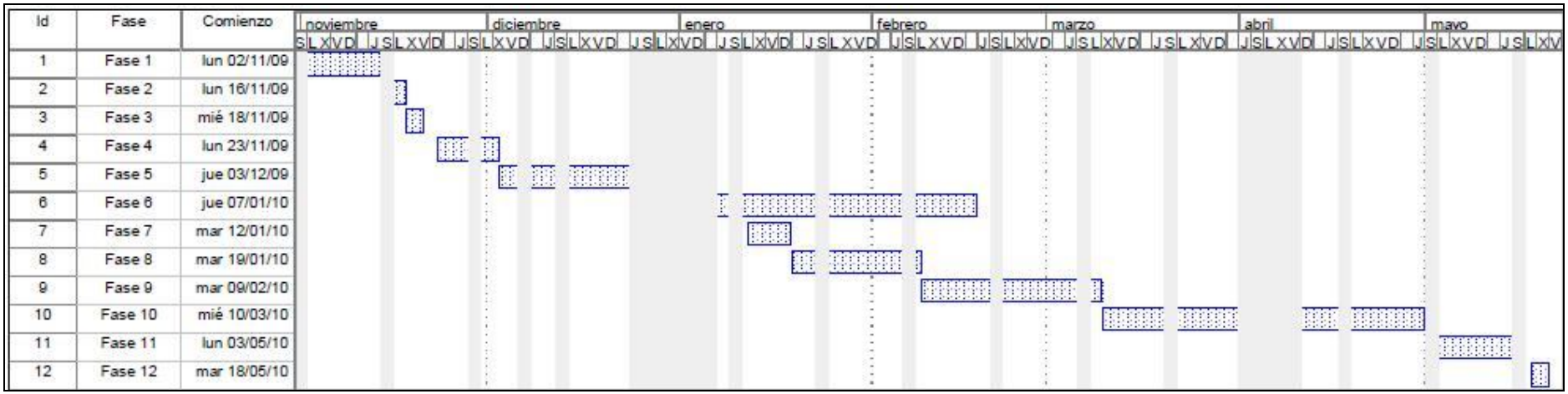
Estimación de esfuerzo: 3 días * 7h/día

Tareas:

- Preparación de la presentación
- Ensayar

Tipo: PowerPoint

2.5 Planificación temporal. Diagrama Gantt:



Id	Fase	Comienzo
1	Fase 1	lun 02/11/09
2	Fase 2	lun 16/11/09
3	Fase 3	mié 18/11/09
4	Fase 4	lun 23/11/09
5	Fase 5	jue 03/12/09
6	Fase 6	jue 07/01/10
7	Fase 7	mar 12/01/10
8	Fase 8	mar 19/01/10
9	Fase 9	mar 09/02/10
10	Fase 10	mié 10/03/10
11	Fase 11	lun 03/05/10
12	Fase 12	mar 18/05/10

Imagen 3 – Diagrama de Gantt

Fase 1: DOP
Fase 2: Estudio del entorno y de las herramientas
Fase 3: Captura de requisitos del subproyecto 1
Fase 4: Análisis del subproyecto 1
Fase 5: Diseño del subproyecto 1
Fase 6: Implementación + Pruebas + Cierre del subproyecto 1
Fase 7: Captura de requisitos del subproyecto 2
Fase 8: Análisis del subproyecto 2
Fase 9: Diseño del subproyecto 2
Fase 10: Implementación + Pruebas + Cierre del subproyecto 2
Fase 11: Memoria
Fase 12: Presentación

2.6 Plan de contingencia:

Para garantizar la continuidad del proyecto, presento una serie de medidas técnicas, humanas y organizativas que prevengan, minimicen o que hagan frente a los riesgos o imprevistos que pudieran surgir.

a. Copias de seguridad:

   Existe el riesgo de perder parte o la totalidad del proyecto. Para hacer frente a este riesgo, dispongo de un disco duro en red del que se hacen backups diarios, en el que se almacena una copia de la documentación relativa tanto a la memoria, como a la generada para la empresa, asegurando así, la integridad y la pronta recuperación en caso de un pérdida total o parcial de la información. Además se harán copias en CD´s cada 15 días y se dispondrá de más copias en diferentes equipos.

b. Retraso en cuanto a la planificación establecida:

   En caso de darse un retraso en cuando a la planificación establecida, se aumentarán el número de horas diarias planificadas para esa tarea, en caso de no ser posible, se aumentará el número de días de trabajo semanal hasta paliar el retraso.

c. Problemas de hardware:

   Para evitar cualquier tipo de retraso a causa del mal funcionamiento del equipo o a causa de problemas hardware, se tendrá preparado un segundo equipo con el software necesario instalado para poder trabajar, en el caso de ser necesario.

d. Desconocimiento de tecnologías:

   En algún momento puede darse una situación no resoluble por incompatibilidad con la herramienta o desconocimiento de la misma por parte del estudiante. Toda acción, operación y/o método debe planificarse teniendo en cuanta que deberá reflejarse en el documento que describa el caso de uso en cuestión una solución alternativa en caso de desconocer como llevarla a cabo en un inicio.

e. Virus informático que altere el normal funcionamiento del equipo:
   Para evitar este posible riesgo, se instalará un antivirus que sea residente en memoria, y se configurará para que chequee continuamente todas las acciones realizadas sobre los archivos. Se actualizará constantemente, ya sea a través de Internet o a través del distribuidor.

f. Privacidad:
   Para preservar la privacidad y la información del proyecto, los equipos que se utilicen estarán protegidos por contraseñas, contraseñas que solo conocerán el administrador y el grupo encargado de redes.

2.7 Factibilidad:

Dado que se dispone aproximadamente de 1200 h/persona para la realización del proyecto y habiendo hecho una estimación de 1000h/persona y teniendo un plan de contingencia que permite abordar los problemas que puedan surgir, considero factible la realización de este proyecto.

3. CAPTURA DE REQUISITOS

3.1 Diagrama de casos de uso:

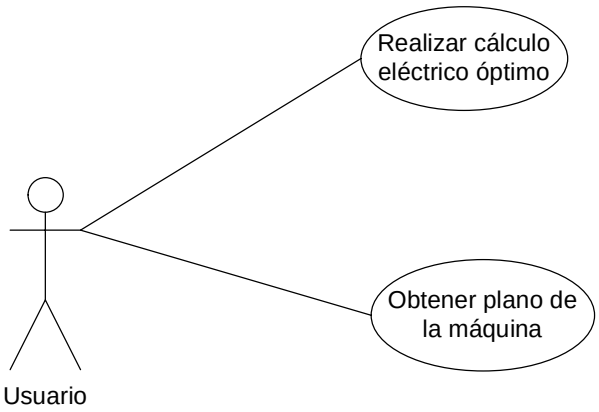


Imagen 4 – Diagrama de Casos de Uso

3.2 Casos de uso:

3.2.1 Realizar cálculo eléctrico óptimo

- Actores: Usuario
- Resumen: Este caso de uso permite realizar el proceso de cálculo eléctrico para la obtención del generador hidráulico óptimo según ciertas características de entrada.
- Precondición: El usuario tiene acceso a la Intranet de la empresa
- Poscondición: Se obtiene el generador hidráulico óptimo según las características definidas por el cliente junto con la especificación eléctrica completa del generador resultante.

Escenario principal (o curso normal de los eventos)

1. Usuario: El usuario quiere realizar el cálculo eléctrico de una máquina.
2. Sistema: Muestra el formulario a rellenar por el usuario
3. Usuario: Rellena el formulario

4. Sistema: Muestra un resumen con los datos introducidos por el usuario junto con el resultado de otros datos calculados con las características de entrada.
5. Usuario: Acepta el cálculo con la especificación que el sistema muestra en pantalla.
6. Sistema: Realiza el cálculo eléctrico.

Extensiones (o cursos alternativos)

4. Sistema: Detecta que alguna de las casillas del formulario no se ha rellenado debidamente o se ha dejado en blanco. Lanza el aviso correspondiente, para que el usuario corrija el error.
5. Usuario: Rellena debidamente el formulario.
6. Sistema: Muestra un resumen con los datos introducidos por el usuario junto con el resultado de otros datos calculados con las características de entrada.
7. Usuario: Acepta el cálculo con la especificación que el sistema muestra en pantalla.
8. Sistema: Realiza el cálculo eléctrico.

3.2.2 Obtener plano de la máquina

- Actores: Usuario
- Resumen: Este caso de uso permite generar planos de oferta con cotas definidas.
- Precondición: El usuario tiene acceso a la Intranet de la empresa.
- Poscondición: Obtenemos un plano de oferta del generador.

Escenario principal (o curso normal de los eventos)

1. Usuario: El usuario quiere obtener el plano de una máquina
2. Sistema: Muestra el formulario a rellenar por el usuario

3. Usuario: Rellena el formulario y acepta la obtención del plano con esas características.
4. Sistema: Genera el plano acorde con las características de entrada proporcionadas por el cliente

Extensiones (o cursos alternativos)

4. Sistema: Detecta que alguna de las casillas del formulario no se ha rellenado debidamente o se ha dejado en blanco. Lanza el aviso correspondiente, para que el usuario corrija el error.
5. Usuario: Rellena debidamente el formulario y acepta el proceso
8. Sistema: Genera el plano acorde con las características de entrada proporcionadas por el cliente

3.3 Modelo de dominio:

Cálculo Eléctrico:

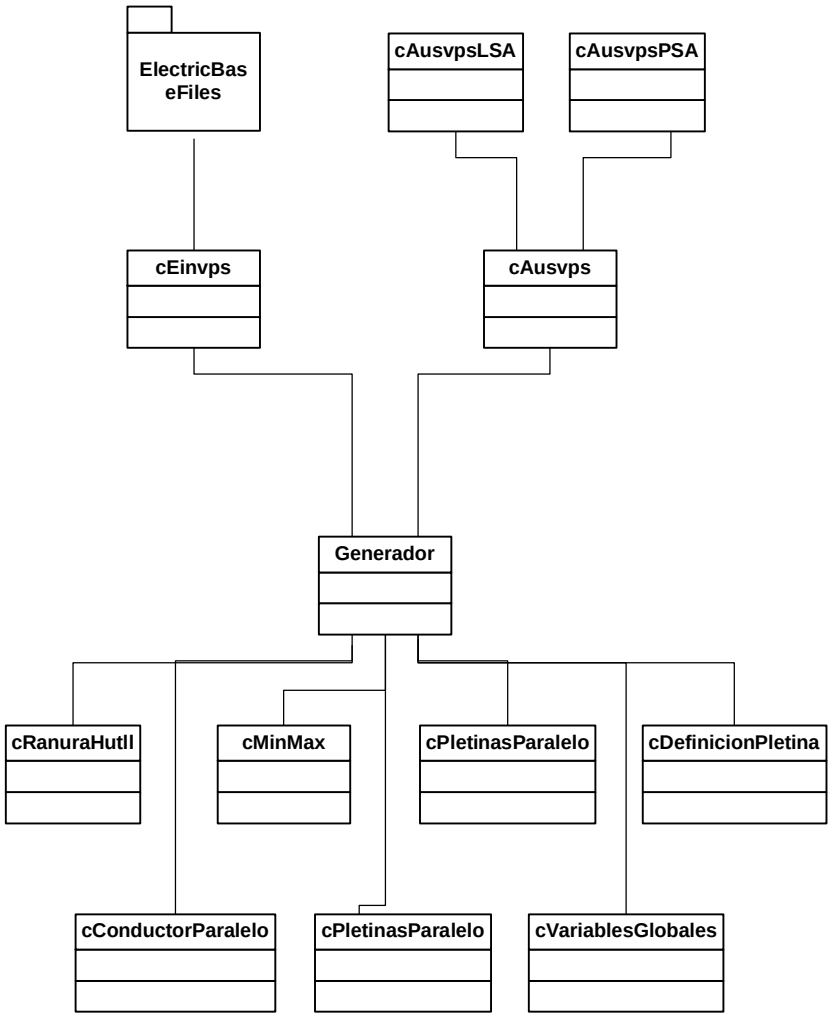


Imagen 5 – Modelo de Dominio: Cálculo Eléctrico

En el siguiente punto procederé a explicar el modelo de dominio. Preservando la privacidad de empresa, mostraré algunos de los atributos más importantes de cada clase, y cual es su función en el proceso.

ElectricBaseFiles es un catálogo de oferta de generadores hidráulicos de Indar. En el podremos encontrar los que denominaré como ficheros base. Los ficheros base están organizados en el catálogo ElectricBaseFiles por altura y número de polos. Los polos son elementos causantes del campo magnético

que se crea entre módulos del generador hidráulico. Si el en catálogo tenemos un fichero base con el nombre 0560.006, significa que INDAR produce generadores hidráulicos de una altura de eje 560 con 6 polos, y a su vez, que el cálculo eléctrico puede ofertar una máquina de este tipo.

Los ficheros base, son archivos de formato .txt creados por el grupo de Técnica eléctrica, personal encargado de todos los aspectos eléctricos de los generadores que se producen. Por cada tipo de máquina que pueda producir la empresa, en el catálogo ElectricBaseFiles tendremos un fichero con la especificación eléctrica base. Para la organización de este tipo de ficheros hemos creado una estructura llamada cEinvps con 112 atributos. Estos son algunos de los más importantes.

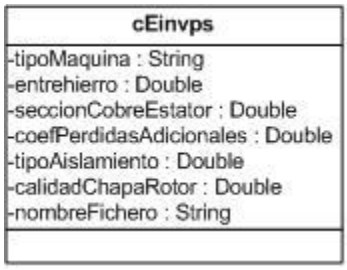


Imagen 6 – Atributos de cEinvps

Para la lectura, modificación y escritura de cEinvps he creado una librería llamada libreríaExterna para el tratamiento de ficheros que explicaré con detalle más adelante.

Los ficheros base son la entrada para la aplicación externa. Esta aplicación es un ejecutable que dado el fichero base como entrada, realiza diversos cálculos eléctricos técnicos y del que obtenemos un resultado a modo de fichero .txt llamado Ausvps.

Ausvps será diferente en función del tamaño de la máquina. Se considera máquina pequeña (LSA), cuando la altura de eje es menor o igual a 1120. Por el contrario, se considera máquina grande (PSA), cuando la altura de eje es mayor que 1120. Para la organización de este tipo de ficheros hemos creado

dos estructuras en función del tamaño de la máquina: cAusvpsLSA con 345 atributos y cAusvpsPSA con 333, estos son algunos de los mas importantes:

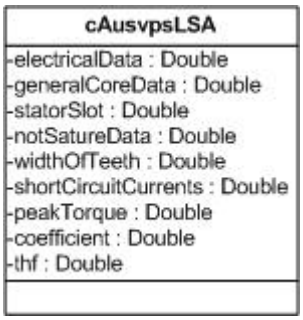


Imagen 7 - Atributos de cAusvpsLSA

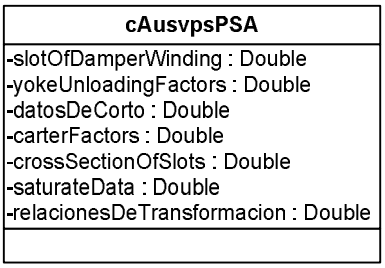


Imagen 8 – Atributos de cAsuvpsPSA

Como he mencionado antes, hemos creado una librería para el tratamiento de los ficheros de entrada, einvps, y para los de salida, Ausvps, para la aplicación externa. Esta librería, llamada LibreríaExterna, será utilizada en el proyecto en multitud de ocasiones, ya que no se llama a la aplicación externa una única vez y el proceso del cálculo eléctrico requiere muchos cambios en dichos archivos.

La razón que nos empujo a realizar un módulo independiente, fue la reutilización, debido a que es muy común la utilización de estos archivos por parte de la empresa. La librería externa consta de tres esqueletos importantes: cEinvps, cAusvpsLSA y cAusvpsPSA, al ser clases con mucha información, cAusvpsLSA y cAusvpsPSA contienen además de atributos, subclases que nos permiten organizar las estructuras de forma más clara.

Las funciones más importantes que maneja la librería externa son:

- Volcar la información contenida en los ficheros .txt einvps y ausvps, en las estructuras correspondientes cEinvps, cAusvpsLSA o cAusvpsPSA.

- Crear ficheros .txt con la estructura exacta a einvps y ausvps volcando las clases cEinvps, cAusvpsLSA o cAusvpsPSA

Por otro lado, las especificaciones del cliente son almacenadas en una clase llamada cDatosIn. Al inicio del proceso, el formulario se carga en esta estructura para que esos datos estén accesibles durante todo el cálculo.

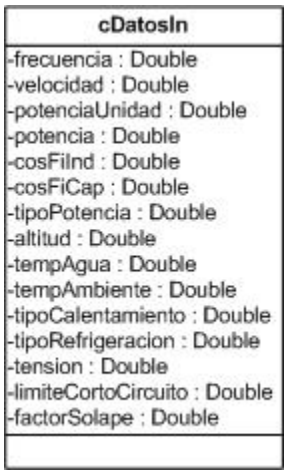


Imagen 9 – Atributos de cDatosIn

La clase generador será la base para nuestro cálculo. Sus atributos serán los parámetros a ajustar para conseguir como resultado el generador hidráulico óptimo. Como sucede con cAusvpsLSA y cAusvpsPSA, la información dentro de la clase se organiza en subclases, tal y como se muestra en el modelo de dominio. Estos son algunos de los atributos más importantes.

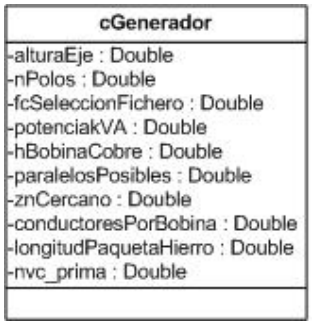


Imagen 10 - Atributos de cGenerador

Generador de planos:

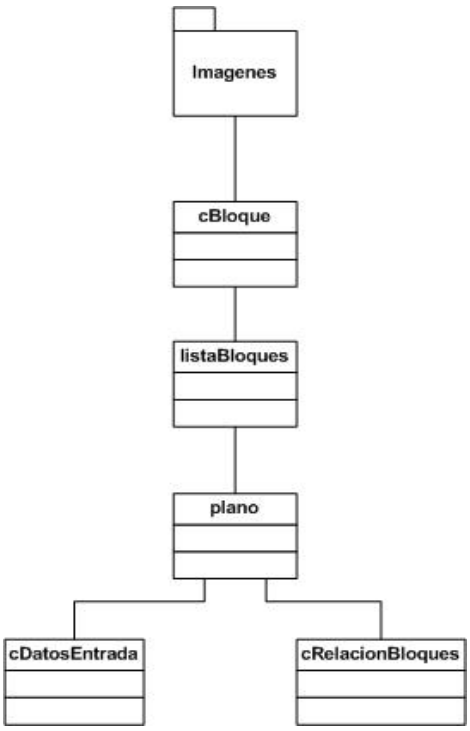


Imagen 11 - Modelo de Dominio: Generador de Planos

El modelo de dominio del Generador de Planos es más simple. El sistema se compone de las clases que se muestran en la imagen 11. La información principal se centra en el Plano. Como en el caso del Cálculo eléctrico, los datos de entrada que proporciona el cliente, se carga en la estructura cDatosEntrada, que estará accesible durante todo el proceso. En la imagen 12 algunos de los atributos más importantes de cDatosEntrada.

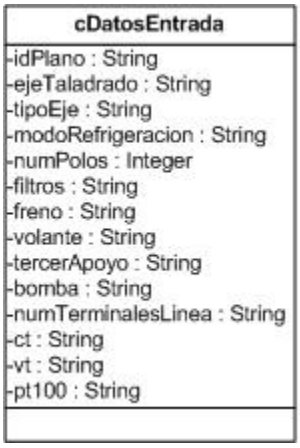


Imagen 12 - Atributos de cDatosEntrada

Se dispone de un catálogo de imágenes en el que encontraremos todas las imágenes necesarias para componer el plano. Tenemos 15 posibles tipos de bloques a insertar en el plano, como se muestra en la siguiente tabla.

DIVISÓN DE BLOQUES	
ID	**NOMBRE**
01	Extremo del eje de LA
02	Apoyo LA
03	Carcasa
04	Rejillas
05	Refrigeración
06	Bancada / Obra Civil
07	Apoyo LO
08	Volante de inercia
09	Rotor LO
10	Tercer Apoyo
11	Caja de Conexiones Potencia
12	Caja de Conexiones Auxiliares
13	Caja de Neutro
14	Grupo Lub. Oil
15	Reacciones Obra Civil

Imagen 13 – Tipos de Bloque

Como se muestra en la siguiente figura, el plano resultante, es un plano 2D que se puede organizar en 5 zonas: la vista A (vista lateral), vista B (vista frontal), imagen miniatura de la vista B, cajetín de información de plano y cajas de texto.

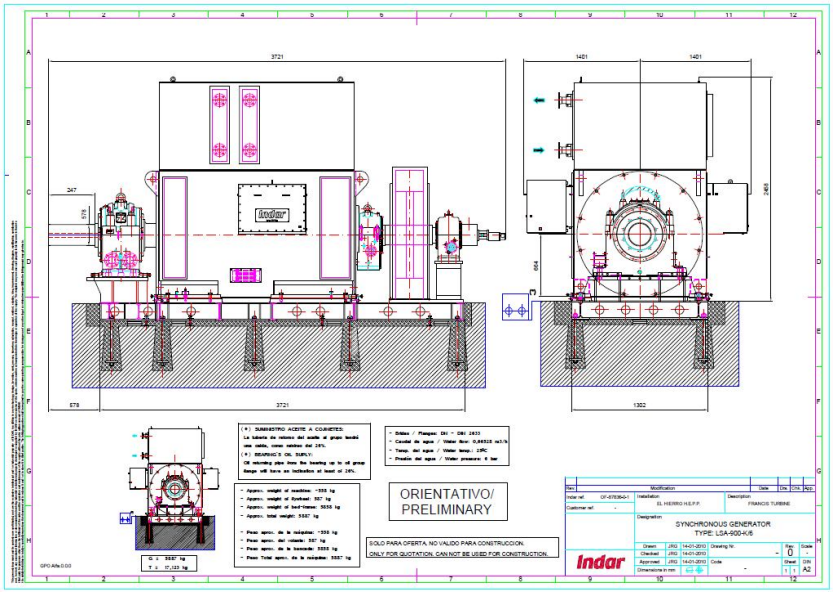


Imagen 14 - Plano resultante

Cada tipo o familia de bloques, dispondrá en el catálogo de tantas imágenes como casuísticas existan para cada vista. A modo de ejemplo, en el caso del bloque de *Rejillas*, con ID 04, dispondremos de las siguientes posibles imágenes:

VISTA A	VISTA B
04.1-A-01	04-B-01
04.1-A-02	04-B-02
04.1-A-03	04-B-03
04.1-A-04	
04.1-A-05	
04.2-A-01	
04.2-A-02	
04.2-A-03	

Tabla 1 – Vistas Rejilla

Un vez que el sistema obtenga la especificación de entrada y mediante una lógica o proceso creado para esté fin, el sistema decidirá según las características de entrada, cuales de los 15 tipos de bloques deberemos insertar para generar el plano y en cada caso, la vistaA y la vistaB correspondiente del catálogo de imágenes.

Por cada tipo de bloque a insertar, se creará un cBloque. Esta estructura, almacenará el nombre del bloque del tipo de bloque y las imágenes vista A y vista B correspondientes del catálogo que el sistema ha determinado para ese tipo de bloque.

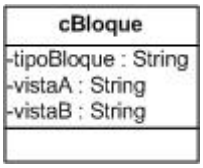


Imagen 15 – Atributos de cBloque

Una estructura cBloque tendría este aspecto:

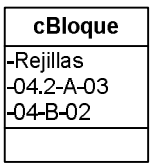


Imagen 16 – Ejemplo cBloque

Contaremos con una listaBloques en la que se irán almacenando todos los cBloques a insertar en el plano.

La clase cRelacionBloques contiene dos atributos, vistaA y vistaB del tipo HashTable. Para entender la labor de esta estructura, es necesario entender cómo es el proceso de inserción de bloques. Al querer que el proceso de inserción de bloques sea automático, no cabía pensar que podíamos decir a cada bloque en que punto del plano debía ser insertado, debido a que no siempre se insertan todos los bloques, los mismos bloques, o en la misma posición. Era necesario buscar la forma de automatizar el sistema. Para ello, se ha ideado una lógica que tiene en cuenta los bloques que se han ido insertado y en que puntos, cuando se inserta un nuevo bloque se consulta toda esta información y la lógica le permite saber el punto de inserción exacto le corresponde a la nueva imagen, esa es la información que se va almacenado en las tablas hash correspondientes a cada vista.

4. ANÁLISIS

Caso de uso: Realizar cálculo eléctrico óptimo

Diagrama de secuencia del sistema:

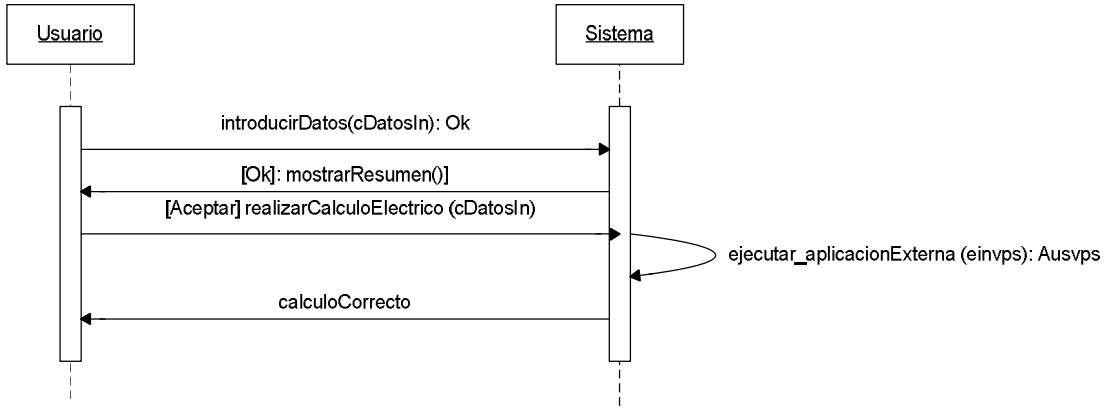


Imagen 17 – Diagrama de secuencia del sistema: Realizar Cálculo Eléctrico

Contratos:

- Name: **introducirDatos (cDatosIn): OK**

- Responsabilities: Esta operación valida los datos de entrada introducidos por el cliente en el formulario. Verifica que el formulario esté correctamente completado, campos obligatorios rellenos, y que cada casilla recoja el tipo de datos correcto.

- Preconditions: El usuario tiene acceso a la Intranet o Extranet de INDAR

- Posconditions: Los datos de entrada del cliente son almacenados en el sistema para que estén disponibles durante todo el proceso

- Salida: En el caso de que el formulario haya sido completado de forma correcta, un valor bolean a True, False en el caso contrario.

- Name: **mostrarResumen()**

- Responsabilities: El sistema muestra un resumen de la especificación del cliente y de unos cálculos previos al Cálculo Eléctrico realizados en base a los introducidos por el cliente.

- Preconditions: El método introducirDatos ha validado el formulario

- Posconditions: El sistema almacena los cálculos previos realizados y da la opción de realizar el Cálculo Eléctrico

- Salida: -

- Name: **[Aceptar] realizarCalculoElectrico (cDatosIn): calculoCorrecto**

- Responsabilities: Es la operación encargada de realizar el proceso de cálculo eléctrico para la obtención del generador hidráulico óptimo según ciertas características de entrada.

- Preconditions: El usuario acepta el Cálculo Eléctrico en base a los datos mostrados por pantalla.

- Posconditions: Los ficheros Einvps, Ausvps y el fichero Log, contendrán información relevante a las características de la máquina seleccionada como generador hidráulico óptimo según las características de entrada del cliente.

- Salida: Un valor bolean que si el cálculo eléctrico es correcto, será True y False en el caso contrario.

Aplicaciones externas utilizadas:

- Name: **ejecutar_aplicacionExterna (einvps): Ausvps**

- Responsabilities: Para realizar el Cálculo Eléctrico se hace uso de una aplicación externa existente creada en exclusiva para la empresa INDAR. Realiza cálculos eléctricos muy técnicos. Como parámetro de entrada, se le proporciona un fichero con la estructura cEinvps, como salida, obtendremos un fichero de la estructura cAusvps. La utilización y estructura de ambos ficheros se detallará mas adelante y en la medida de lo posible, siempre preservando la privacidad de la empresa.

Caso de uso: Obtener plano de la máquina

Diagrama de secuencia del sistema:

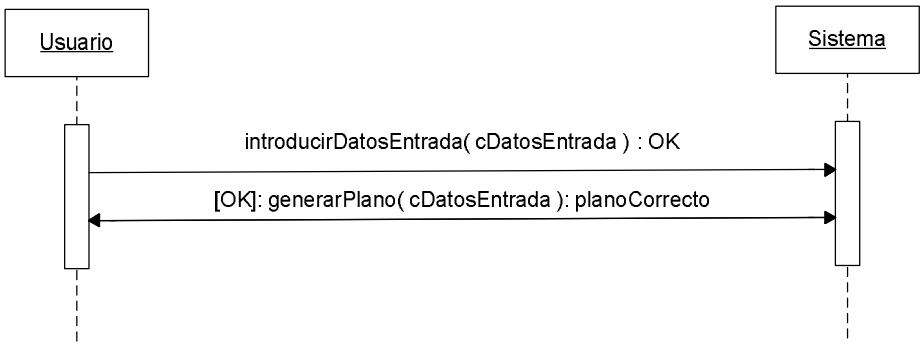


Imagen 18 - Diagrama de secuencia del sistema: Obtener plano de la máquina

Contratos:

- Name: **introducirDatosEntrada (cDatosEntrada): OK**

- Responsabilities: Esta operación valida los datos de entrada introducidos por el cliente en el formulario. Verifica que el formulario esté correctamente completado, campos obligatorios rellenos, y que cada casilla recoja el tipo de datos correcto.

- Preconditions: El usuario tiene acceso a la Intranet o Extranet de INDAR

- Posconditions: Los datos de entrada del cliente son almacenados en el sistema para que estén disponibles durante todo el proceso

- Salida: En el caso de que el formulario haya sido completado de forma correcta, un valor bolean a True, False en el caso contrario.

- Name: **[OK] generarPlano(cDatosDeEntrada): planoCorrecto**

- Responsabilities: Es la operación encargada de generar los planos de oferta con cotas definidas.

- Preconditions: El método introducirDatosEntrada ha validado el formulario

- Posconditions: Obtendremos el plano resultante en formato pdf y dwg[1]

- Salida: Un valor bolean que si el plano se ha generado correctamente, será True, en caso contrario, False.

[1] Extensión de archivo electrónico de dibujo computarizado, utilizado principalmente por el programa AutoCAD; producto de la compañía AutoDesk

5. ARQUITECTURA

5.1 Arquitectura del sistema:

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. En cuanto a patrones de arquitectura se refiere, dos son los más comunes:

- Modelo vista controlador:
    - Modelo: Es el encargado de la lógica de negocio y el acceso a la base de datos.
    - Vista: Interacciona con la interfaz de usuario.
    - Controlador: Responde a eventos, generalmente del usuario e invoca cambios en el modelo y en la vista.

- Arquitectura en tres capas:
    - Capa de presentación: Es la capa que ve el usuario, presenta el sistema al usuario. Interacciona únicamente con la capa de negocio. Es la encargada de presentar los resultados generados por capas inferiores.
    - Capa de negocio: Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. Se encarga de la funcionalidad de la aplicación.
    - Capa de datos: Capa encargada de la gestión de los datos, recibirá consultas y devolverá datos a la capa de negocio.

Tras mucho meditarlo, nuestra elección ha sido el patrón arquitectónico en 3 capas.

La capa de presentación de cada aplicación, consta de un único formulario a rellenar por el usuario y que permitirá lanzar el proceso completo mediante el botón de aceptar.

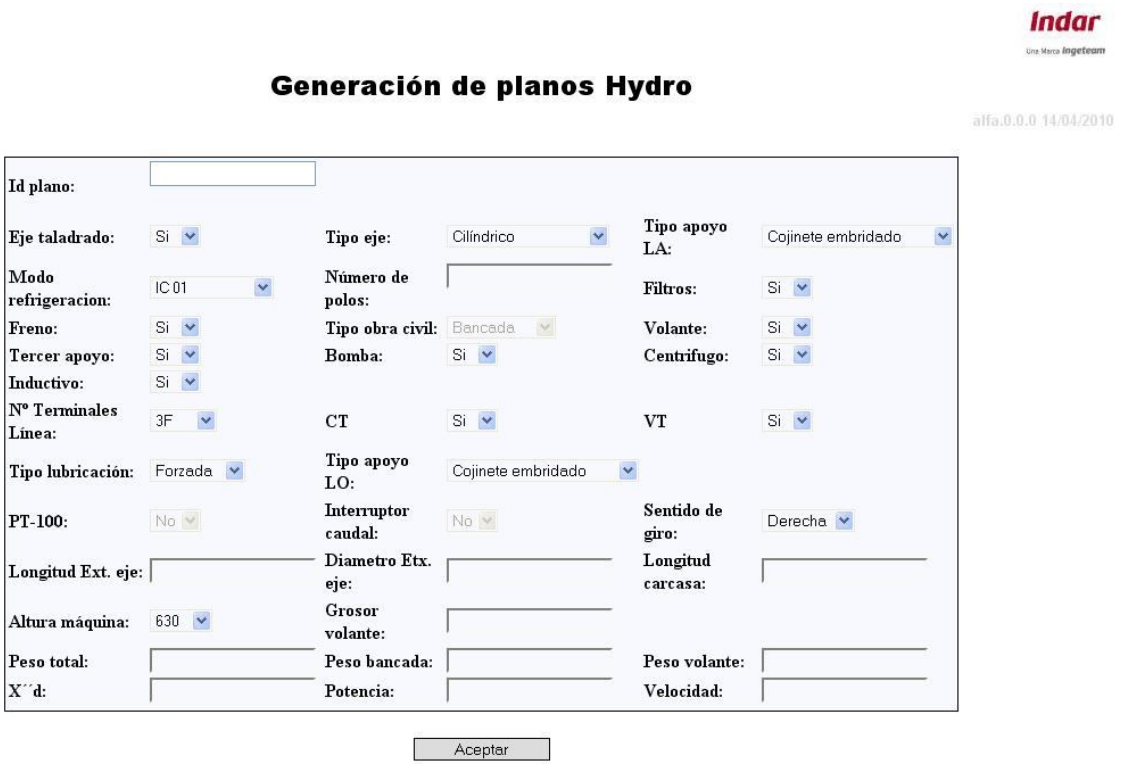


Imagen 19 - Formulario Generación de Planos

Para la gestión de datos, crearemos una base de datos que recibirá consultas y devolverá datos a la capa de negocio. Los datos que no estén incluidos en la base de datos, serán extraídos de ficheros txt propiedad de INDAR. Para la gestión de este tipo de ficheros se ha creado una librería llamada libreríaExterna que se encargará de cargar los datos en estructuras que serán tratadas desde la capa de negocio.

5.2 Elección tecnológica:

La elección tecnológica no ha estado en mi mano, debido a que venía determinado por la empresa. Para el desarrollo del proyecto hemos utilizado Visual Basic su entorno de trabajo:

- Visual Web Developer 2008 Express con Framework 3.5 para el desarrollo Web y diseño de los formularios.

- Visual Basic 2008 Express Edition para el desarrollo de librerías

Imagen 20 - Visual Basic 2008 Express Edition

Para la gestión de los datos, hemos optado por Microsoft SQL Server 2005 Express Edition.

La aplicación para Generar Planos requería de un software de diseño. La empresa limito las opciones a dos: AutoCAD 2010 y Solid Edge, ya que disponía licencias para ambos paquetes. Después de contemplar las dos opciones y probar su funcionalidad, nos decantamos por AutoCAD 2010 ya que Solid Edge estaba enfocado a diseño 3D.

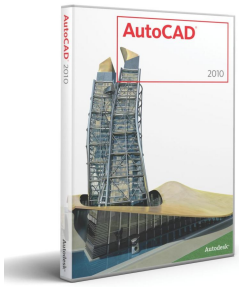


Imagen 21 - AutoCAD 2010

6. DISEÑO

A continuación detallaré el diseño de cada uno de los subproyectos.

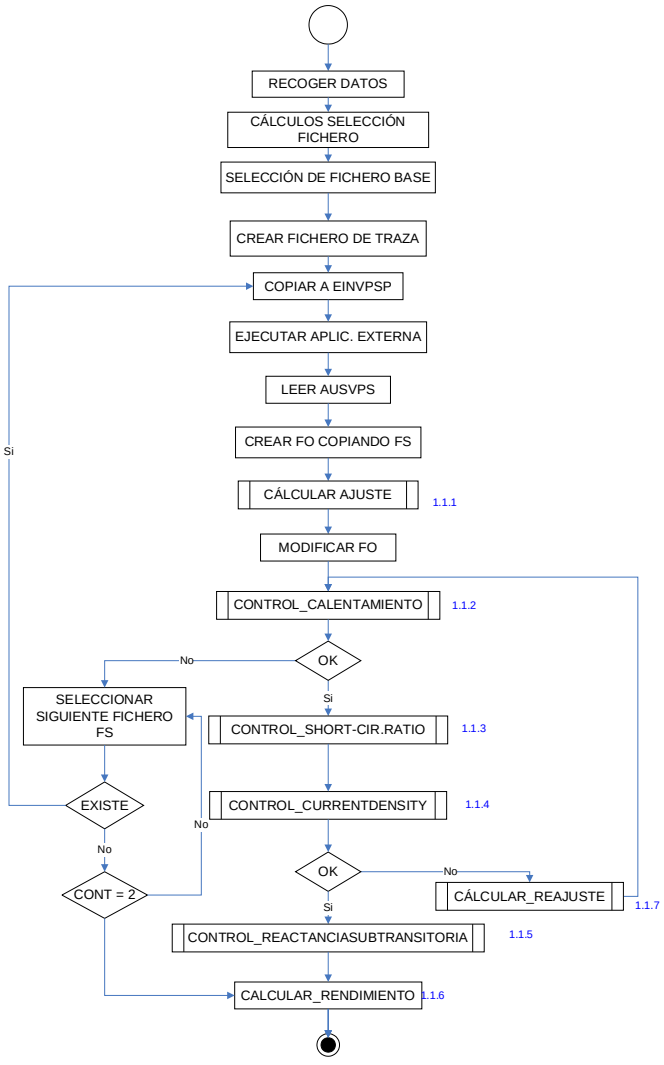
Con el fin de preservar la privacidad y al tratarse de aspectos eléctricos muy técnicos no detallaré en que consisten cada uno de los cálculos realizados en cada punto del proceso.

6.1 Cálculo Eléctrico:

Con el objetivo de facilitar la comprensión añadiré diagramas que muestran el flujo de la ejecución en las operaciones que lo requieran. El diseño detallado será una breve explicación del proceso que sigue el método.

6.1.1 Realizar Cálculo eléctrico:

- Diagrama de flujo:



Algoritmo 1 – Realizar Cálculo Eléctrico

▪ Diseño detallado:

• Recoger los datos proporcionados por el cliente en el formulario

• Seleccionar un fichero base (FS) del catálogo ElectricBaseFiles. Al estar los ficheros base organizados por número de polos y altura de eje, previo a la selección del fichero, realizaremos cálculos como: el número de polos, potenciakVA, factor corrector, potencia control solape, límite de calentamiento, etc. Después, podremos seleccionar el fichero base que se ajusta más a las características de entrada.

• Crear un fichero de traza. El objetivo de este fichero es reflejar el proceso del cálculo de las características relevantes, para que los Analistas de Técnica eléctrica puedan revisarlos en caso de necesitarlo.

• Copiar el fichero base a einvps que será la entrada para la aplicación externa.

• Ejecutar la aplicación externa que tendrá como entrada una copia del fichero base (FS) seleccionado al inicio del proceso.

• Leer Ausvps. Ausvps es la salida que obtenemos de la aplicación externa. Recogeremos los datos del .txt mediante la libreríaExterna, los trasladaremos al fichero de traza y los almacenamos en la estructura correspondiente cAusvpsLSA o cAusvpsPSA. Estos datos los necesitaremos posteriormente.

• Copiar el fichero base seleccionado (FS) a un fichero “oferta” (FO) que será el archivo sobre el que se realizarán las modificaciones requeridas por el ajuste o el reajuste.

• *Calcular ajuste()*

• Modificar el fichero oferta (FO) con los datos introducidos por el cliente, para ello, utilizaremos las funciones creadas en la librería para la lectura y escritura de ficheros de entrada einvps y el de salida ausvps.

• Modificar el fichero oferta (FO) con los datos obtenidos al realizar el ajuste de los parámetros de la máquina, para ello, utilizaremos las funciones creadas en la librería para la lectura y escritura de ficheros de entrada einvps y de salida ausvps

• Mientras no fin

fin = true

motivo = *Control de calentamiento()*

If motivo = 0 //El control de calentamiento es correcto y la máquina con las características que muestra hasta el momento, no se calentará

Control de Short Circuit ratio()

maquinaCorrecta = *Control de Current Density 2()*

If maquinaCorrecta = true //El control de curren density es correcto, es el último control de calentamiento. Llegados a este punto podemos afirmar que hemos encontrado nuestro generador óptimo y que cumple todos los requerimientos impuestos por normas de Indar.

Control de Reactancia Subtransitoria()

Calcular rendimiento()

Escribir resumen de resultados en el fichero de traza

Calcular el largo de la máquina y especificarlo como resultado en el fichero Log, einvps y ausvps, y que estos 3 ficheros forman el resultado de la aplicación.

Else

fin = false //Para seguir en el while

Calcular reajuste()

End If

End If

Fin mientras

If motivo = 1 // Si control de calentamiento devuelve un 1 significa que con el fichero base que tenemos no hemos conseguido un generador que no se caliente y se ha optado por elegir el siguiente fichero base, mismo número de polos pero siguiente altura de eje.

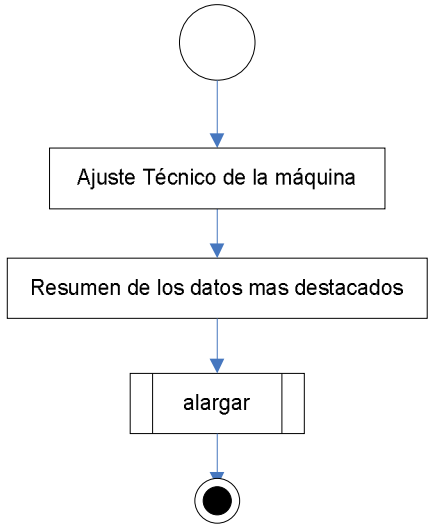
Realizar cálculo eléctrico() // Recursividad

End if

A continuación, explicaré la función de las operaciones más importantes del Cálculo Eléctrico como son: El ajuste eléctrico de una máquina, el control de calentamiento, el control de short circuit ratio, el control de current density, el control de reactancia subtransitaria, el cálculo de rendimiento y el cálculo de reajuste.

6.1.2 Calcular ajuste:

- Diagrama de flujo:



Algoritmo 2 – Calcular ajuste

- Diseño detallado:

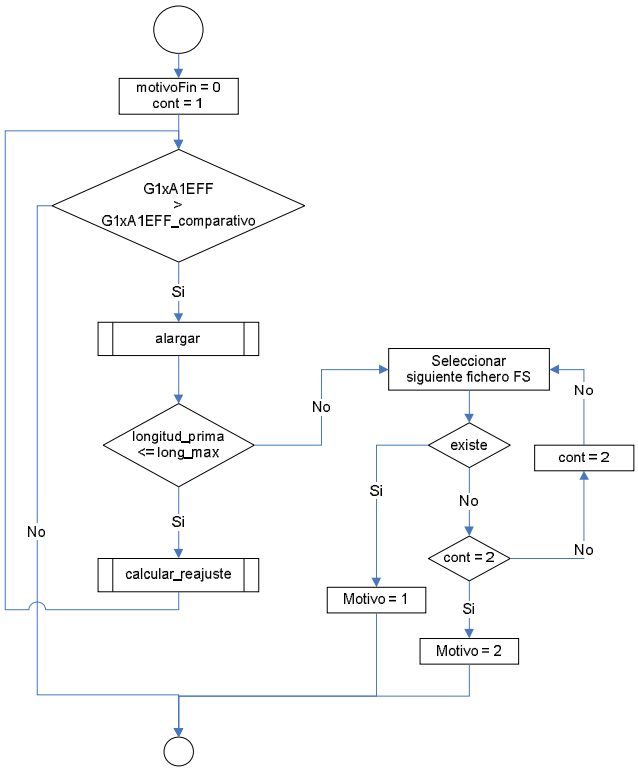
Este método ajusta aspectos eléctricos de la máquina con el fin de que sean viables y lo mas ajustados posible a los requerimientos del cliente.

Se realizan cálculos como: número de paralelos posibles, conductores en serie por ranura, dimensión de la pletina, altura de la bobina de cobre, espacio util en ranuras... Dichos cálculos han sido trabajados y especificados por el grupo de Técnica eléctrica de la sección Hydro de la empresa INDAR.

Un vez realizado el ajuste de los aspectos eléctricos de la máquina se muestra un resumen del estado actual de la máquina en el fichero de traza.

6.1.2 Control de calentamiento:

▪ Diagrama de flujo:



Algoritmo 3 – Control de Calentamiento

▪ Diseño detallado:

Este método controla que el calentamiento de una máquina no exceda de los límites establecidos. Si el calentamiento sobrepasa dicho límite, se procede a alargar la máquina hasta que el valor de calentamiento sea mas bajo que el límite. Cada máquina del catálogo de ofertas de Indar, dispone de diferentes longitudes. Al alargar la máquina, obtendremos un valor de calentamiento mas bajo.

Si no disponemos de la siguiente longitud para poder alargar nuestro tipo de máquina, se procederá a elegir la siguiente altura de eje, eligiendo para ello el siguiente fichero base con el mismo número de polos, pero con la siguiente altura.

Posibles salidas:

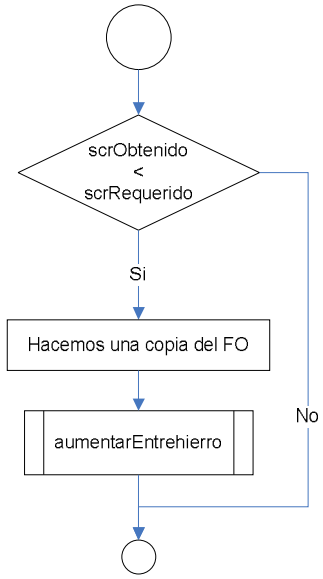
motivoFin = 0: Se a alargado la máquina

motivoFin = 1: Se ha elegido la siguiente altura de eje

motivoFin = 2: Por norma Indar, si las siguientes dos alturas de eje no están disponibles para nuestro tipo de máquina, se para el proceso porque se ofrecería al cliente una máquina mucho mas grande de la que debería en cuanto a las especificaciones iniciales establecidas.

6.1.3 Control de Short Circuit Ratio:

▪ Diagrama de flujo:



Algoritmo 4 – Control de Short Circuit Ratio

▪ Diseño detallado:

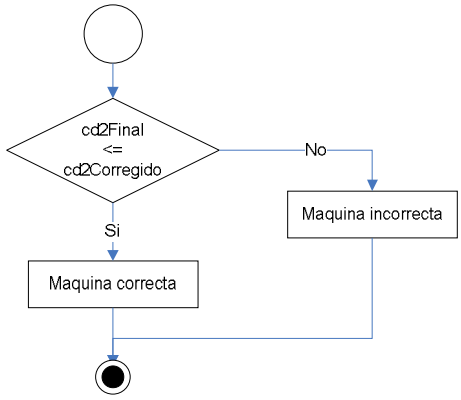
El límite de cortocircuito[2] es un parámetro que puede venir establecido por el cliente (scrRequerido), mientras el límite de corto circuito que nos proporciona el generador (scrObtenido) sea inferior al requerido por el cliente, se procederá a aumentar el entrehierro.

El entrehierro es el espacio libre que hay entre el estator y el rotor de un generador. El aumento del entrehierro proporcionará mayor límite de corto circuito a la máquina.

[2] Se denomina **cortocircuito** al fallo en un aparato o línea eléctrica por el cual la corriente eléctrica pasa directamente del conductor activo al neutro o tierra

6.1.4 Control de Current Density 2:

▪ Diagrama de flujo:



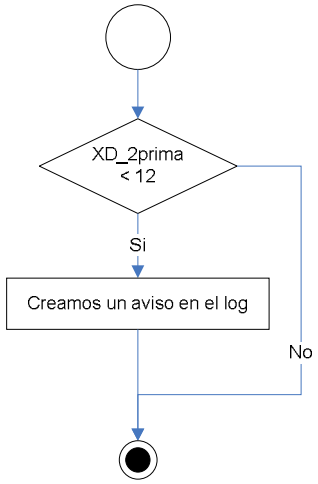
Algoritmo 5 – Control Current Density 2

▪ Diagrama de flujo:

Este método realiza otro control de calentamiento. Esta vez, se calculan los Amperios/milimetro^2 que viajan por el cobre del rotor de la máquina. Si ese valor sobre pasa el límite impuesto por Indar, la maquina no se puede dar por buena ya que el cobre se calentaría más de lo debido.

6.1.5 Control de Reactancia Subtransitoria:

▪ Diagrama de flujo:



Algoritmo 6 – Control de Reactancia Subtransitoria

▪ Diseño detallado:

El grupo de Técnica eléctrica de la unidad de trabajo Hydro, solicitó el control de reactancia subtransitoria. Dicho control, como muestra el diagrama, se trata de una única comparación. Si el dato XD_2Prima es menor que 12 deberemos crear un aviso en el fichero Log para que Técnica eléctrica lo tenga en cuenta.

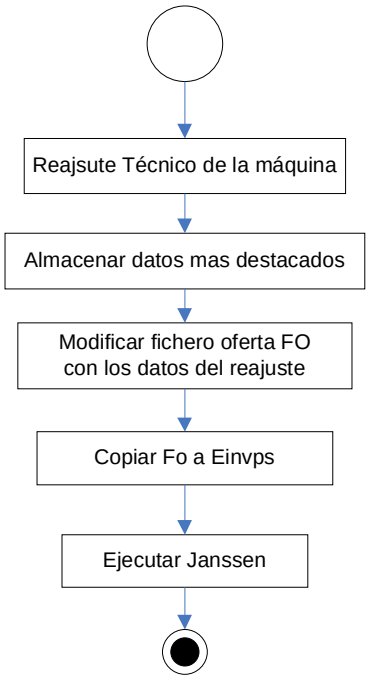
6.1.6 Cálculo de rendimientos:

▪ Diseño detallado:

El rendimiento de un generador está estrechamente ligado a la potencia que entrega a la red. Cuanto mayor es el rendimiento, mayor es la potencia entregada y mayor es el coste de la máquina. Mediante esta función se cálcula cual es el rendimiento del generador resultante del proceso.

6.1.7 Calcular reajuste:

▪ Diagrama de flujo:



Algoritmo 7 – Calcular reajuste

- Diseño detallado:

  Este método reajusta aspectos eléctricos de la máquina con el fin de que sean viables y lo más ajustados posible a los requerimientos del cliente.

  Se realizan cálculos como: combinaciones de conductores, sección efectiva de la máquina, selección de pletina, control de alturas normalizadas... Dichos cálculos han sido trabajados y especificados por el grupo de Técnica eléctrica de la sección Hydro de la empresa INDAR.

  Después del reajuste eléctrico se procederá a:

  - Almacenar los aspectos mas destacados del cálculo en el fichero de traza.
  - Modificar el fichero oferta (FO) con los resultados del cálculo eléctrico.
  - Copiar el fichero oferta a einvps (fichero entrada para la aplicación externa)
  - Ejecutar aplicación externa

6.2 GENERAR PLANOS:

Dada la dificultad del proceso de generación de planos, todo lo mostrado a continuación es una simplificación del proceso real.

Con el fin de preservar la privacidad y al tratarse de aspectos mecánicos internos no detallaré todo el proceso a seguir para generar el plano resultante.

Para poder entender la lógica que seguimos en generador de planos, pensaremos que el plano es una especie de mecano a construir. Disponemos de una plantilla sobre la que se insertan los distintos tipos de bloques en los puntos correspondientes. Las carcasas, tanto en la vista A como en la vista B, vienen incluidas en la plantilla, ya que se trata de bloques fijos que no dependen de los datos de entrada.

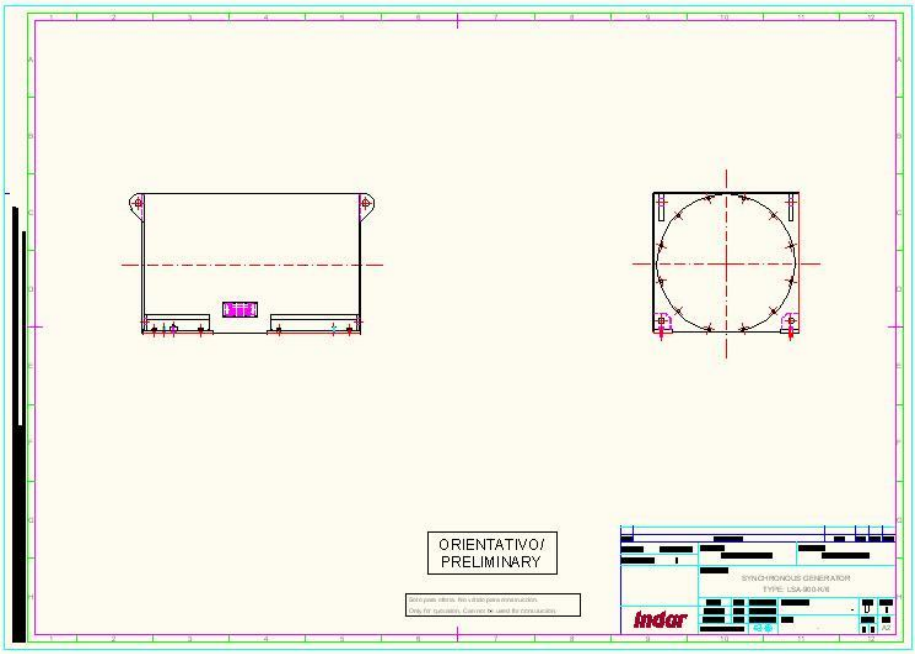


Imagen 22 - Plantilla

Como he mencionado anteriormente, jugaremos con 15 tipos de bloques y con sus diferentes casuísticas para cada vista, que obtendremos del catálogo de imágenes e iremos insertando dependiendo de los datos de entrada proporcionados por el cliente.

La dificultad añadida de esta aplicación es que no siempre se necesitan todos los bloques, no todos los tipos de bloques se inserta siempre en el mismo punto, por lo que no podemos prefijar el punto de inserción de todas las imágenes y además, el aspecto que pueden presentar dos imágenes del mismo tipo de bloque puede ser diferente. A modo de ejemplo, estas son dos de las posibles imágenes del tipo de bloque Rotor LO para la vista A, disponibles en el catálogo.

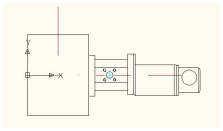


Imagen 23 - Rotor LO 01

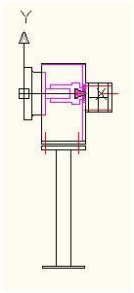


Imagen 24 - Rotor LO 02

Después de estudiar la forma de automatizar el proceso de inserción de las imágenes, dividimos el problema en dos: por un lado, los bloques que en caso de ser requeridos por los datos de entrada, su punto de inserción era fijo y por el otro lado, los bloques que dependían de la existencia de otros bloques para definir su punto de inserción. Para ellos definimos la siguiente tabla en la base de datos. Esta tabla se carga en la estructura cRelacionBloques al inicio del proceso y será utilizada hasta la finalización del mismo.

Tabla 2 – Relación bloques vista A

	bloque	inserción
	01-A	02-A
	02-A	A
	04.1-A	B
	04.2-A	E
	05.1-A	H
	05.2-A	I
	05.3-A	J
	06-A	G
	07-A	F
	08-A	07-A
	09-A	07-A
	09-A	08-A
	09-A	10-A
	10-A	08-A
	11-A	D
	12-A	C

La tabla 2 define la relación para la inserción de bloques en la vista A, tendríamos otra tabla similar para la vista B.

Los tipos de bloques que son visibles desde la vista A son los que se muestran en la columna bloque, por lo que se deduce que el tipo de bloques 15 (Reacciones Obra Civil), 14 (Grupo Lub.Oil), etc, no son visibles desde la vista A.

De esta tabla extraeremos la dependencia entre bloques para su posterior inserción. Los bloques que tienen como punto de inserción una letra, son tipos de bloques que se colocan sobre las carcasas incluidas en la plantilla. Para facilitar la labor de inserción cada carcasa de cada vista, tiene definidos unos puntos de inserción, que en el caso de la vista A, son los siguientes:

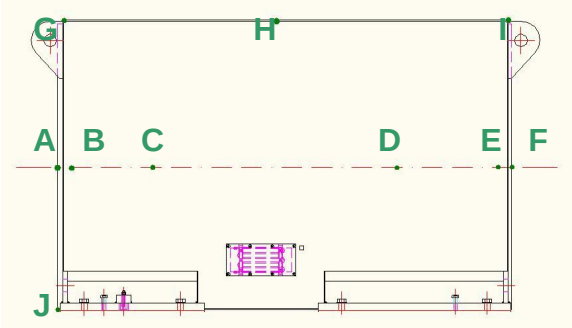


Imagen 25 – Puntos de inseción vista A

Una vez que el sistema, mediante lógicas y consultas a la base de datos, decide que imagen del catálogo corresponde al tipo de bloque a insertar, se consulta en la estructura cRelacionBloques cual es su punto de inserción y se extraen las coordenadas exactas desde la carcasa.

Los bloques que no tienen un punto de inserción prefijado, es porque dependen de la existencia o posición de otro tipo de bloque, como se muestra en la tabla anterior. En este caso, se obtendrá el bloque del que dependen, y se realizarán cálculos para definir en punto exacto de inserción del nuevo bloque.
En la siguiente imagen se muestra un resultado del Generador de Planos, con algunos de los tipos de bloques posibles.

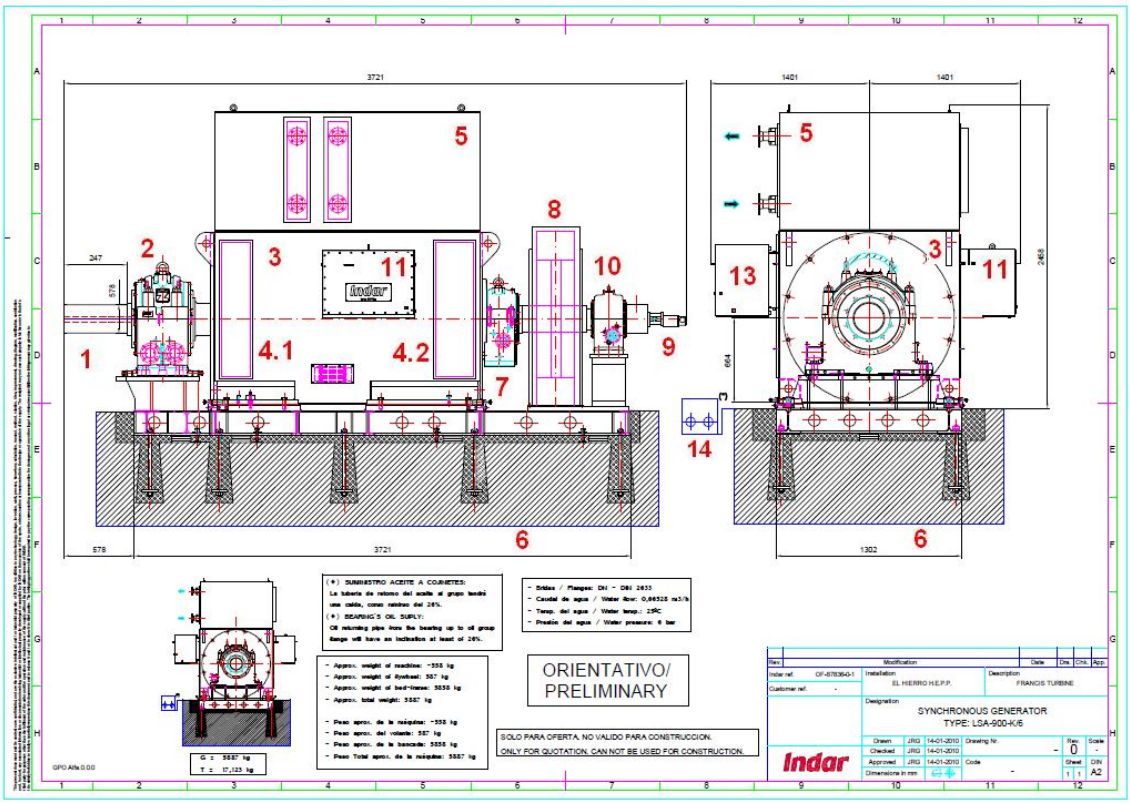
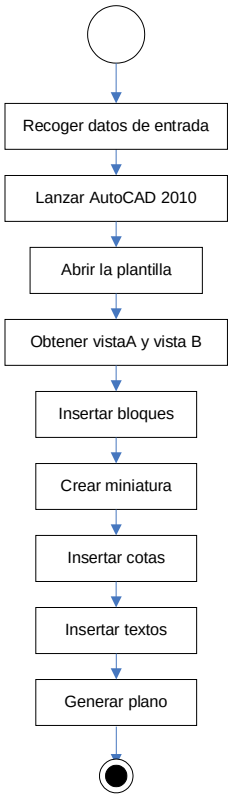


Imagen 26 – Tipos de bloque en Plano resultante

6.2.1 Generar Plano:

- Diagrama de flujo:



Algoritmo 8: Generar Plano

▪ Diseño detallado:

• Recoger los datos de entrada introducidos por el cliente en el formulario

• Lanzar el AutoCAD 2010

• Abrir el archivo que contiene la plantilla mediante AutoCAD.

• El sistema, siguiendo la lógica especificada por la sección de Ingeniería Mecánica de la división Hydro, determinará que tipos de bloques necesitamos, y de cada tipo de bloque, la vista A y la vista B correspondiente del catálogo de imagenes.

• Insertar sobre la plantilla una copia de todo el conjunto de bloques que forman la vistaB, escalarlo y colocarlo en la esquina inferior izquierda.

• Insertar las cotas necesarias para cada caso. Los valores de las cotas serán operaciones entre datos extraídos de la base de datos y datos robados de los cálculos realizados por el Configurador de Ofertas.

• Insertar textos. Tenemos 5 cajas de texto a insertar: refrigeración, lubricación, pesos, reacciones y obra civil. La inserción de textos también sigue una lógica definida. No todos los textos tienen que aparecer siempre. Los valores, como en el caso de las cotas serán operaciones realizadas entre datos extraidos de la base de datos y datos robados de los cálculos realizados por el Configurador de Ofertas.

• Centrar y escalar el documento AutoCAD resultante para que el plano se ajuste a las dimensiones dinA4. Una vez esté centrado y

escalado, lo convertimos a pdf lanzando el conversor de AutoCAD, “DWG to pdf.pc3” y guardamos el .pdf en la ruta especificada.

• Almacenar el plano resultante (formato .dwg) en la ruta especificada

• Cerrar los AcadDocument utilizados

• Cerrar AutoCAD

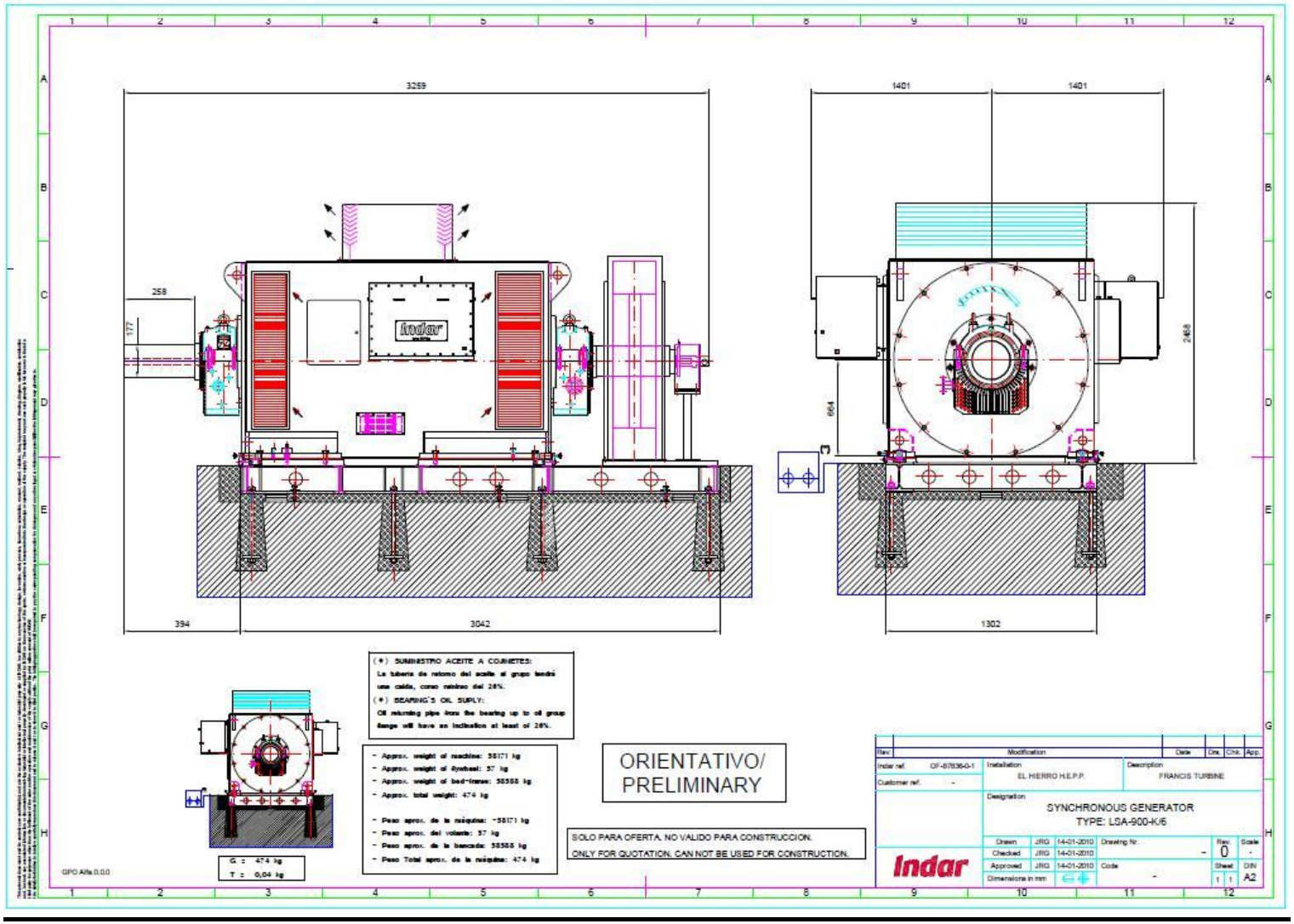


Imagen 27 - Generador de Planos: Plano resultante

7. IMPLEMENTACIÓN

Por política de privacidad de la empresa en la que he desarrollado el proyecto, no es posible adjuntar el código de las aplicaciones. Por ello explicaré a grandes rasgos, cómo se ha realizado la implementación de dicha aplicación. He adjuntado parte del código, bien por ser totalmente nuevo para mí, o bien por la sencillez con la que he conseguido afrontar aspectos que consideraba complicados, buscando las herramientas adecuadas.

Las aplicaciones se han implementado con Visual Basic y los entornos de trabajo han sido Visual Web Developer 2008 Express, para el desarrollo Web y Visual Basic 2008 Express Edition, para el desarrollo de las librerías.

7.1 Cálculo Eléctrico:

7.1.1 System.Reflection:

Una vez recopilada y entendida la información que teníamos que manejar para poder sacar adelante este proyecto, nos encontramos con el siguiente problema. Teníamos que poder tratar ficheros .txt con mucha información. Para ello, tuvimos que crear un conjunto de clases con cientos de atributos cada una. Teníamos que poder leer, modificar y crear nuevos ficheros .txt con la misma estructura que el original, volcando los datos de las clases. Conscientes del tiempo que nos podía llevar crear los métodos para leer, modificar y escribir de cada clase, tratando cada elemento uno por uno, optamos por dedicar un tiempo a buscar una alternativa y nos encontramos con *System.Reflection.*[3]

[3] El espacio de nombres System.Reflection contiene clases e interfaces que proporcionan una vista administrada de los campos, los métodos y los tipos cargados, con la posibilidad de crear e invocar tipos dinámicamente.
http://msdn.microsoft.com/es-es/library/system.reflection%28VS.80%29.aspx

A continuación, mostraré un ejemplo del uso de **System.Reflection**:

Este es el aspecto que muestra un fichero einvps, que sirve de entrada a la aplicación externa existente en la empresa.

```
     'N.A.                                          ' 'LSA-150-X/3
' 1.00800000 1 1.00000000 3.00000000 0 0 1 2 14.5000000 14.5000000
0.05000000 1.00000000 4.00000000 25.5000000 4.00000000 25.5000000
3.00000000 1.90000000 9.00100000 2.50000000 3.50000000 69.0000000
10.0000000      0.00000000E+00      0.00000000E+00      -1.00000000
0.00000000E+00   0.00000000E+00   0.00000000E+00   0.00000000E+00
0.00000000E+00   0.00000000E+00   0.00000000E+00   0.00000000E+00
16.5000000 3.00000000 10.0000000 8.00000000 960.000000 700.000000
370.000000 620.000000 5.00000000 72.0000000 1.00000000 0.899999976
1  6000.00000  0.00000000E+00  1400.00000  50.0000000  3.00000000
0.00000000E+00  1.70000005  0  72.0000000  12.0000000  10.0000000
0.00000000E+00  0.00000000E+00  75.0000000  29.5000000  19.0000000
19.0000000 0.00000000E+00 75.0000000 49.5000000 1 0.00000000E+00
0.349999994 0 1.00000000 0.949999988 0.00000000E+00 0.00000000E+00
0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00  0
1.00000000 10.0000000 3.15000010 12.0000000 1.00000000 1.00000000
1  0.00000000E+00  1.0000000  1.00000000  54.0000000  10.0000000
400.000000 600.000000 57.0000000 64.0000000 12.0000000 9.00000000
9.00000000  1.00000000  4.50000000  0.00000000E+00  4.900005000
0.00000000E+00 1.70000005 1.00000000 1.00000000 0.00000000E+00
      '0150.03                        '
```

Este es el fichero mas pequeño que tenemos que ser capaces de manejar, contiene 112 datos a almacenar en una estructura. Como he comentado con anterioridad, se ha creado una librería que contendrá los métodos para tratar los datos de este tipo de ficheros. Procederé a explicar a grandes rasgos el método creado para leer un fichero einvps, centrándome en la importancia del uso de Reflection.

- leer_einvps: Se encarga de leer de un fichero einvps y volcarlo en la estructura cEinvps.

Utilizando la herramienta *StreamReader* leemos un fichero ubicado en la ruta pasada como parámetro y obtenemos su contenido en un String.

```
Dim sr As StreamReader = New StreamReader (ruta)
Dim linea As String
linea = sr.ReadLine
sr.Close ()
```

Con ayuda de la función *Split*, cortamos el String `linea` por el delimitador, que en este caso, será el espacio, obteniendo un array que contendrá en cada posición un atributo de cEinvps.

```
datosLinea.datos = Split(lineaDatos, " ")
```

Una vez tengamos todos los datos de einvps en un array, utilizaremos Reflection para volcarlo en la clase creada con anterioridad con los 112 atributos. Reflection nos permite recorrer todas las propiedades de un objeto, en este caso cEinvps, como si fuera un bucle y asignarles un valor. Cabe mencionar que en el caso de cEinvps, casi todos los atributos son de tipo Double, pero hay 3 que son de tipo String, por lo que a la hora de recorrer las propiedades de cEinvps habrá que detenerse en esos casos y tratarlos de forma especial.

```
Dim propiedad As PropertyInfo
Dim null As Object() = Nothing
Dim cont As Integer = 0

Try

    For Each propiedad In einvps.GetType.GetProperties()

        If (propiedad.Name = "nombre") Then
            propiedad.SetValue(einvps, datos.nombre, null)

        ElseIf (propiedad.Name = "tipoMaquina") Then
            propiedad.SetValue(einvps, datos.tipoMaquina, null)

        ElseIf (propiedad.Name = "nombreFichero") Then
            propiedad.SetValue(einvps, datos.nombreFichero, null)

        Else
            If cont < datos.datos.Length Then

                propiedad.SetValue(einvps, CDbl(Replace(datos.datos(cont), ".", ",")), null)
                cont = cont + 1
            End If

        End If

    Next
Catch ex As Exception

    MsgBox("Error: guardar_datosEnEinvps" & vbCrLf & ex.ToString)

End Try
```

Como se muestra en el ejemplo, se recorren los atributos de einvps (del tipo cEinvps), almacenándolos en “propiedad”, independientemente del tipo que sea (String, Double…).

En muy pocas líneas de código conseguimos realizar un trabajo que en cualquier otro caso, hubiera resultado muy tedioso. Esa es la dinámica que hemos utilizado para el tratamiento de ficheros con multitud de datos.

7.1.2 Objetos Application y Session:

Ahora nos vamos a centrar en la necesidad por parte de la empresa de crear sesiones simultáneas a la aplicación que realiza el cálculo eléctrico del generador óptimo. Uno de los problemas de la ejecución simultánea, sería el acceso a la aplicación externa. Imaginemos que los usuarios 1 y 2 han lanzado el cálculo electrico en el mismo instante, el usuario 1 llega al punto del código en el que se ejecuta la aplicación externa, y cuando procede a recoger los datos resultantes, el usuario 2 ha ejecutado también la aplicación externa y ha modificado el archivo de salida ausvps, machacando los resultados de la ejecución del usuario 1. Este es uno de los muchos problemas que evitaremos controlando las ejecuciones simultaneas, para ello, se crearán variables de aplicación y de sesión.

.NET nos proporciona una clase **HTTPApplicationState** por cada uno de nuestros sitios web. Se trata de una estructura del tipo diccionario de claves-valor, cualquier valor que guardemos estará disponible desde cualquier página del sitio.

El **objeto Application** se crea en el momento en el que el primer usuario accede a nuestro sitio web, y se destruye en el momento en el que el último usuario lo abandona. Cualquier información guardada en el objeto application, estará disponible para todos los usuarios de nuestro sitio web.

Por ejemplo, un contador del número de usuarios del sitio. Siguiendo con el ejemplo del contador del número de usuarios del sitio, la variable de aplicación se inicializará a 0 una única vez. Cuando el sitio web se ponga en funcionamiento por primera vez y cada vez que un usuario accede al sitio, se accederá a la variable y se incrementará su valor.

El **objeto Session** por el contrario, se crea en el momento en el que un usuario accede a nuestro sitio web, y se destruye en el momento en el que el usuario lo abandona. Cualquier información guardada en el objeto session, estará disponible para un único usuario de nuestro sitio web. Por ejemplo un carrito de la compra en un sitio de comercio electrónico.

Una variable de aplicación se define dentro del archivo **Global.asax.vb** que se encuentra en el directorio App_Code**,** en la raíz de nuestro sitio Web. Dentro de este archivo tenemos un método llamado **Application_Start** y es en este método donde daremos el valor a nuestras variables de aplicación. Todas las aplicaciones Web cuentan con un archivo Global.asax, el cual nos permite ejecutar código asociado a eventos a nivel de nuestra aplicación, por ejemplo el inicio de la aplicación o el inicio de cada una de las sesiones

En el caso que se nos presenta, crearemos una única variable de aplicación que será una lista de posibles aplicaciones externas a ejecutar. Si queremos posibilitar 4 accesos simultáneos crearemos en nuestro proyecto 4 carpetas iguales con diferente nombre que contendrán todo lo que necesitamos para ejecutar la aplicación externa. La variable de aplicación tendrá 4 entradas que definirán cuales de las 4 posibles ejecuciones están en uso y cuales libres. Se actualizará cada vez que un usuario bloquee una aplicación externa y cada vez que lo libere. Esta variable será única para toda la aplicación.

La razón que nos obliga a optar por esta opción es que la aplicación externa que usamos con frecuencia en nuestro proceso, es una aplicación cerrada. Nos obliga a tener el fichero de entrada einvps al mismo nivel

que la propia aplicación externa y que a su vez genera el fichero de salida Ausvps también al mismo nivel, dificultando las ejecuciones simultáneas.

Todos los usuarios de un sitio web pueden acceder a una determinada pareja de clave-valor en el objeto Application, lo que puede conllevar problemas de concurrencia. Para evitarlos, siempre que guardemos un valor en el objeto Application, lo bloquearemos antes y lo desbloquearemos después (exclusión mutua).

```
Application. Lock ()
Application (“Clave”) = valor
Application. Unlock ()
```

El método gestiona las sesiones simultaneas se encarga de que una vez el usuario tenga ocupado una de las 4 accesos simultáneos a la aplicación externa, la variable de aplicación correspondiente se modifica a ocupado, si otro usuario lanza la aplicación, el método le asignará una ejecución “libre”. Una vez el usuario haya acabado con la aplicación que ha ocupado, al final del cálculo, se libera.

7.1.3 XML:

Tanto en el Cálculo Eléctrico, como en la Generación de Planos, al tratarse de aplicaciones que generan resultados a modo de ficheros (.dwg, .pdf, .txt), era importante incluir en la carpeta resultante de la ejecución los datos de entrada proporcionados por el cliente. Para almacenar dicha información, se optó por el formato XML

Este es el aspecto que muestran los ficheros .xml creados a partir de los datos de entrada del formulario.

```
<DatosIn>

 <potencia>3000</potencia>
```

```
 <potenciaUnidad>kW</potenciaUnidad>
 <tipoPotencia>En eje</tipoPotencia>
 <tension>690</tension>
 <frecuencia>60</frecuencia>
 <velocidad>1200</velocidad>
 <cosFiCap>0.95</cosFiCap>
 <cosFiInd>0.9</cosFiInd>
 <factorSolape>0.7</factorSolape>
 <tempAmbiente>40</tempAmbiente>
 <altitud>900</altitud>
 <tempAgua>25</tempAgua>
 <tipoCalentamiento>Clase B</tipoCalentamiento>
 <tipoRefrigeracion>IC 01</tipoRefrigeracion>
 <limiteCortoCircuito>1</limiteCortoCircuito>
 <horizontalVertical>Horizontal</horizontalVertical>

</DatosIn>
```

El fomulario está organizado en tablas, <asp:Table>, las tablas en filas, <asp:TableRow>, y las filas, en celdas <asp:TableCell>. Para recorrer los controles del formulario deberemos acceder a las celdas de cada fila para cada tabla del formulario. Dependiendo de si es un RadioButton, un TextBox o un Label, obtendremos el control, crearemos un nodo con el Id del control, le asignaremos el valor y lo añadiremos al documento Xml.

Después de crear un xml con los datos del formulario de entrada, se vuelca dicho archivo en la estructura de datos correspondiente, cDatosIn. Para poder utilizar Reflection y poder volcar el xml sin recorrer uno por uno los atributos de la clase, los ID de los controles del formulario y las propiedades de la clase cDatosIn tienen nombres idénticos, de esta forma, una vez tengamos el nodo del Xml, obtenemos el ID y el valor de ese ID, y le asignamos el dicho valor a la propiedad de cDatosIn con el mismo nombre.

```
Public Function volcarXml (ByVal ruta As String) As cDatosIn

Dim reader As XmlTextReader = New XmlTextReader (ruta &
    "XML.xml")
    Dim c As cDatosIn = New cDatosIn
    Dim nombre, valor As String
    Dim tipo As Type
    Dim null As Object () = Nothing

    While (reader.Read ())

        Select Case reader.NodeType
            Case XmlNodeType.Element
                'El nodo es un elemento.
                nombre = reader.Name
                Exit Select

            Case XmlNodeType.Text
                'Muestra el texto en cada elemento
                valor = reader.Value
                tipo = c.GetType.GetProperty (nombre).GetValue(c,
                                                null).GetType
                c.GetType.GetProperty (nombre).SetValue(c, valor,
                                            null)
                Exit Select
        End Select
    End While

    Return c

End Function
```

7.2 Generador de planos:

7.2.1 VB.NET para AutoCAD 2010:

Uno de los retos a los que me he enfrentado en la realización de este proyecto, ha sido, "controlar" una herramienta muy potente como es **AutoCAD 2010** desde VB.net. Desde VB.NET debíamos manejar bloques dibujados en AutoCAD 2010, ir insertandolos sobre un documento AutoCAD nuevo, acotar bloques, introducir textos, centrar el plano resultante escalándolo para que ocupase las dimensiones de un Din A4 y ser capaz de convertir a .pdf. Todo ello nos ha llevado mucho tiempo de búsqueda de información en libros, en foros, tutoriales…

Después de mucho tiempo, hallamos todas las respuestas que necesitabamos. Voy a explicar a grandes rasgos, todo lo necesario para crear una aplicación VB.NET que interactue con AutoCAD 2010.

El primer paso es añadir la librería de clases AutoCAD. Añadiremos AutoCAD 2010 Type Library e importaremos la librería AutoCAD en nuestro proyecto.

- Abrir AutoCAD: Hay dos formas para abrir AutoCAD, siendo visible para el usuario o siendo invisible.

```
Dim objApp As New AcadApplication
objApp.Visible = True
```

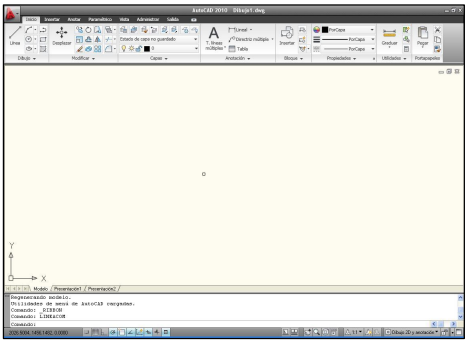


Imagen 28 - AutoCAD lanzado desde VB.NET

- Crear un documento AutoCAD y añadirle el documento a la aplicación AutoCAD abierta.

```
Dim objDoc As AcadDocument
objDoc = objApp.Documents.Add()
```

- Crear un bloque, almacenar en esa variable bloque, una pieza que tenemos dibujada en un .dwg y a su vez insertarla en el documento AutoCAD en una posición determinada y con unas carcterísticas definidas.

```
Dim objBlock As AcadBlockReference
objBlock = objDoc.ModelSpace.InsertBlock (insertpt,
                                    strFileName,1,1,1,0)
```

insertpt: punto de inserción del bloque

strFileName: ruta al bloque a insertar

- Para insertar una cota, utilizaremos los siguientes objetos, dependiendo de la dirección que toma la cota:

```
Dim cot As AcadDimRotated
Dim cot2 As AcadDimAligned
```

Habrá que definir los extremos de la cota, el grado de rotación, posición, tamaño, contenido, estilo y punto en el que colocar el texto.

- Centrar el plano resultante e imprimirlo a pdf utilizando la propia herramienta *DWG to pdf.pc3* de AutoCAD 2010. Configuraremos el objeto ActiveLayout según nuestras especificaciones. En nuestro caso, hacemos una ventana de selección entre los puntos vPt1 y vPt2, lo centramos y lo escalamos, después, convertimos a .pdf utilizando el

método PlotToFile de Plot, pasando como parámetro la ruta destino y el nombre de la herramienta capaz de pasar a .pdf.

```
Public Sub centrarPlanoAPdf (ByVal objDoc As AcadDocument,
     ByVal rutaDestino As String)

          Dim vPt1 (0 To 1) As Double
          Dim vPt2 (0 To 1) As Double

          vPt1 (0) = 0
          vPt1 (1) = 0
          vPt2 (0) = 594
          vPt2 (1) = 420

          With objDoc.ActiveLayout
              .SetWindowToPlot (vPt1, vPt2)
              .PlotType = AcPlotType.acWindow
              .CenterPlot = True
              .StandardScale = AcPlotScale.acScaleToFit
          End With
          objDoc.Plot.PlotToFile (rutaDestino & "prueba.pdf",
                                  "DWG to pdf.pc3")
End Sub
```

El resultado es un archivo .pdf centrado y escalado

7.2.2 JavaScript:

Otro aspecto a mencionar, es la utilización de **JavaScript**, para validar formularios Html. JavaScript, es un lenguaje con muchas posibilidades, utilizado para crear pequeños scripts que luego son insertados en una página Web. Con JavaScript podemos crear diferentes efectos e interactuar con nuestros usuarios.

Tomaremos como ejemplo, la casilla número de polos del formulario de Generación de Planos. Es un campo obligatorio, utilizaremos el control de validación RequiredFieldValidator. Los controles de validación nos

permiten automatizar las tareas de revisión de la información introducida por el usuario en un formulario, tales como comprobar que un campo tiene algún dato, o que ese dato cumple una serie de condiciones.

```
<asp:RequiredFieldValidator ID="RFV_numPolos" runat="server"
Text="*" errorMessage="Número de polos es un campo obligatorio"
ControlToValidate = "numPolos" />
```

Los controles de validación incorporados a nuestros formularios no se verán hasta que nuestro usuario intente enviar el formulario sin haber introducido toda la información que le requeríamos.

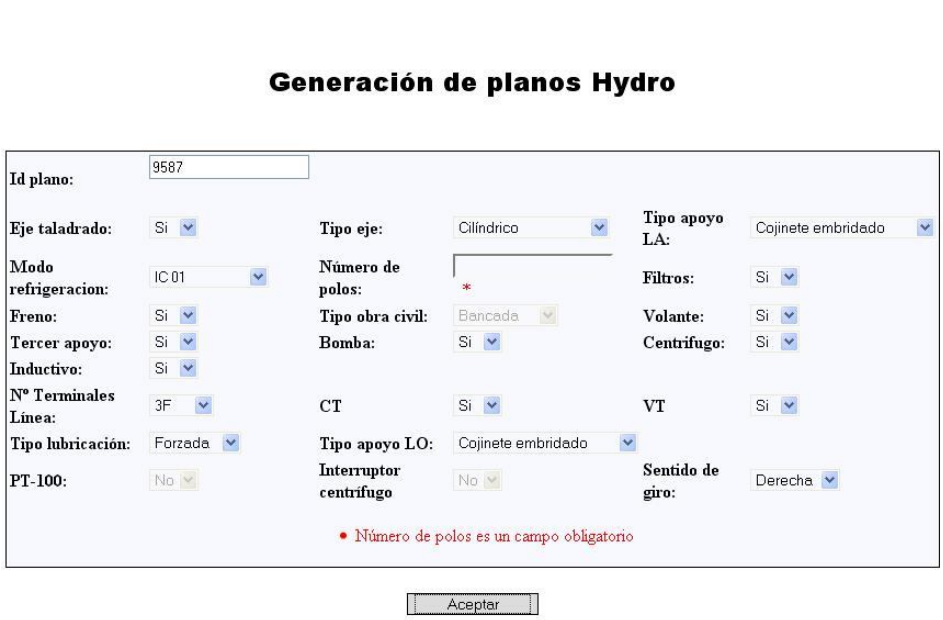


Imagen 29 – Validación de formularios 01

El número de polos también tiene que ser un número par. El control CustomValidator, nos permite emplear funciones de validación específicas que ya tengamos creadas, asignándolas a la propiedad ClientValidationFunction.

```
<asp:CustomValidator ID = "CV_numPolos" runat = "server" Text =
"*" ErrorMessage= "El número de polos, tiene que ser un número
par" ControlToValidate = "numPolos" ClientValidationFunction =
verificarNumPar(); ></asp:CustomValidator>
```

La función *verificarNumPar* es la función JavaScript que se muestra debajo, se coloca dentro de la etiqueta <body></body> de la página html. En caso de que el valor introducido en la casilla numero de polos, sea impar, se borra el contenido de la misma dejando la casilla vacía y se lanza una alerta haciendole saber al usuario, que el número de polos tiene que ser un número par o no podrá seguir adelante con el proceso.

```
<script type="text/javascript" language="javascript">

    function verificarNumPar (){

        var numPolos = document.getElementById('numPolos').value
        var polos = parseInt(numPolos);
        if (polos > 0) {
                if (polos % 2 != 0) {
                    document.getElementById('numPolos').value = ""
                    alert("El número de polos tiene que ser par");
                }
        } else {
            alert("El número de polos tiene que ser mayor que 0");
        }
    }
</script>
```

Imagen 30 – Validación de formularios 02

También se utilizan funciones JavaScript para bloquear algunos de los desplegables, dependiendo de las opciones elegidas en otras casillas, evitando así configuraciones de máquina no procedentes que lleven a la aplicación a fallar.

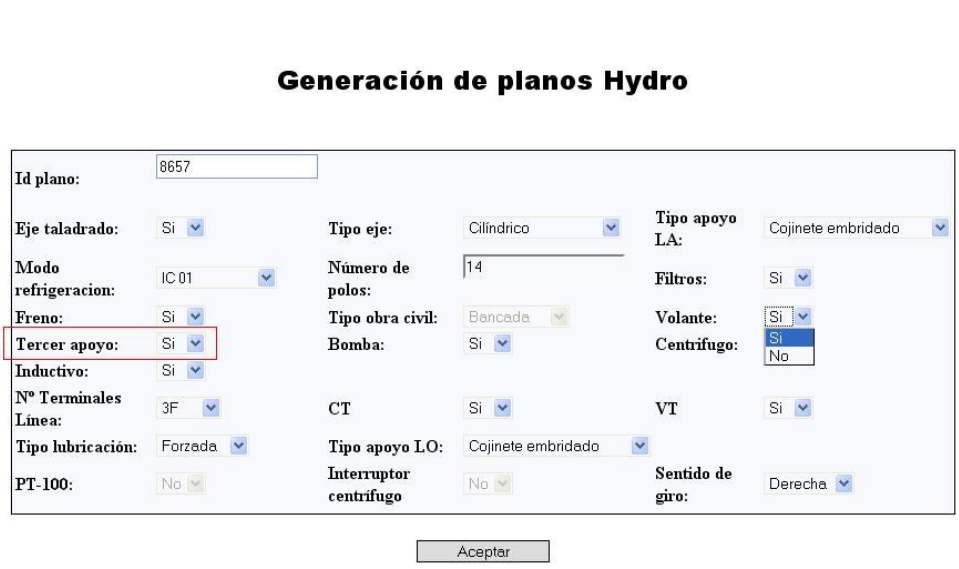


Imagen 31 - Uso del JavaScript en formularios 01

En la figura de arriba, podemos ver, desplegables como *Tipo de obra civil, PT-100 e Interruptor centrifugo* bloqueados. Vamos a ver el comportamiento del formulario, sobre *Tercer apoyo* cuando seleccione "No" sobre el desplegable de *Volante.*

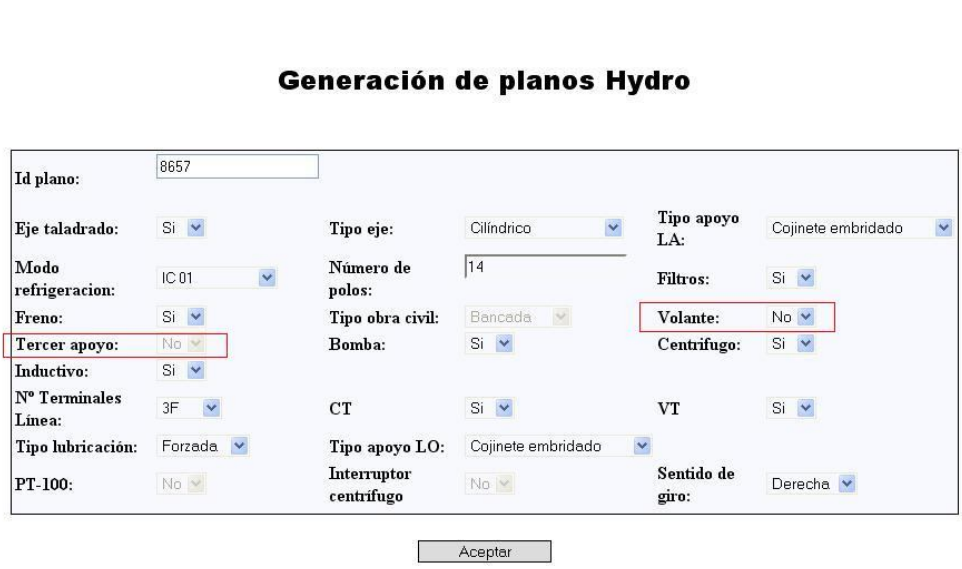


Imagen 32 - Uso del JavaScript en formularios 02

Si no hay Volante, no se puede dar el caso de que haya un Tercer Apoyo, es por lo que no permitimos que se pueda seleccionar, bloqueando la opción en No al usuario.

Es importante recalcar que al hacer estas verificaciones en el código HTML mediante JavaScript, la recarga de la página se hace en el propio cliente, evitando así que sea el servidor el que realice ese trabajo.

7.2.3 ShellAndWait:

Cabe destacar también la utilización de la función Shell para la ejecución de la aplicación externa, debido a que es una aplicación pesada, que necesita un tiempo para ejecutarse, se optó por crear la función **ShellAndWait**, que se encarga de esperar a que la ejecución termine para seguir con el proceso, evitando así conflictos al intentar recuperar la salida generada por dicho programa.

7.2.4 Creación de fichero de traza:

Por último, mostraré la función utilizada para crear el **fichero Log** que se anexará en la carpeta resultante de la aplicación Cálculo Eléctrico.

Inicialmente se comprueba si el fichero ya existe en la ubicación pasada como parámetro, en caso de existir, se borra, después, se crea un nuevo fichero y se escribe en el, la fecha y hora, para que ese archivo sea único y no haya confusiones.

```
Public Shared Sub logFileCreate(ByVal ruta As String)

        If File.Exists(ruta) Then
            File.Delete(ruta)
        End If
```

```
            Dim fs As FileStream
            fs = File.Create(ruta)
            fs.Close()
            fs = Nothing

            Dim sw As New System.IO.StreamWriter(ruta, True)
            sw.WriteLine(DateTime.Now.ToString)
            sw.WriteLine("")
            sw.Close()


    End Sub
```

8. PRUEBAS

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. Una de las cosas que más nos ha sorprendido de esta fase, es la cantidad enorme de tiempo y esfuerzo que requiere. Se estima que la mitad del esfuerzo de desarrollo de un programa, tanto en tiempo como en gastos, se va en esta fase, dado que los fallos de software ocasionan grandes pérdidas económicas.

Estas son los términos que más se utilizan en esta fase:

- Prueba: Actividad en la cual se somete a un sistema o a uno de sus componentes a una evaluación de los resultados que arroja en base a la ejecución de éste en condiciones especificadas.

- Caso de prueba: Conjunto de entradas y condiciones que arrojan resultados esperados desarrollados con un objetivo en particular. Un buen caso de prueba es aquel que tiene una alta probabilidad de descubrir un error no encontrado hasta entonces.

- Error: Acción humana que produce o genera un resultado incorrecto.

- Defecto: Es la manifestación de un error en el software.

- Verificación: Determinar si los productos de una fase dada satisfacen las condiciones impuestas al inicio de la fase. ¿Estamos construyendo correctamente el producto?

- Validación: Evaluación de un sistema o uno de sus componentes durante o al final de su desarrollo para determinar si satisface los requisitos. ¿Estamos construyendo el producto correcto?

La prueba ideal de un sistema sería exponerlo a todas las situaciones posibles, así encontraríamos hasta el último error, garantizando de esta forma su respuesta ante cualquier caso que se presente en una ejecución real. En la práctica, esto es imposible desde todos los puntos de vista: humano, económico.

Someteremos al proyecto a una serie de pruebas con el fin de: encontrar y subsanar fallos existentes, consiguiendo prevenir al sistema del mayor número de fallos posible. Lograremos también confianza acerca del nivel de calidad del producto que hemos creado, afirmando que funciona correctamente y según los requisitos del cliente. Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente.

El desarrollo de este proyecto está basado en ciclo de vida en cascada, como muestra la siguiente imagen.

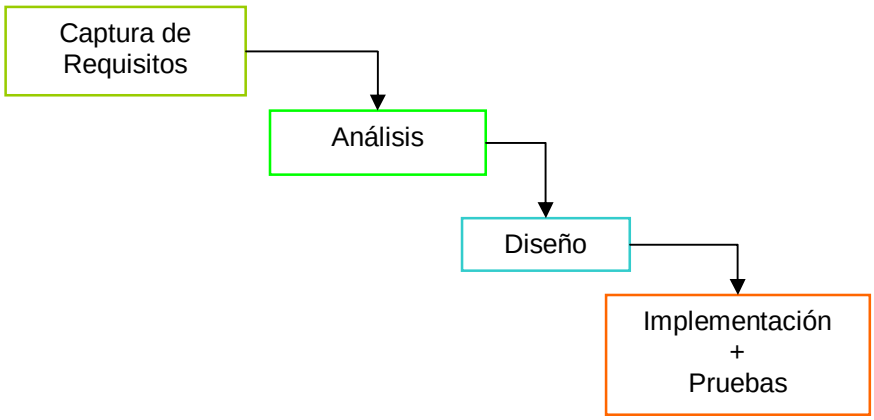


Imagen 33 - Desarrollo en cascada

Estrechamente ligado al modelo en cascada está el modelo en V, puesto que es una evolución del mismo. Este modelo involucra chequeos de cada una de las etapas del modelo de cascada una vez terminado el proceso de codificación.

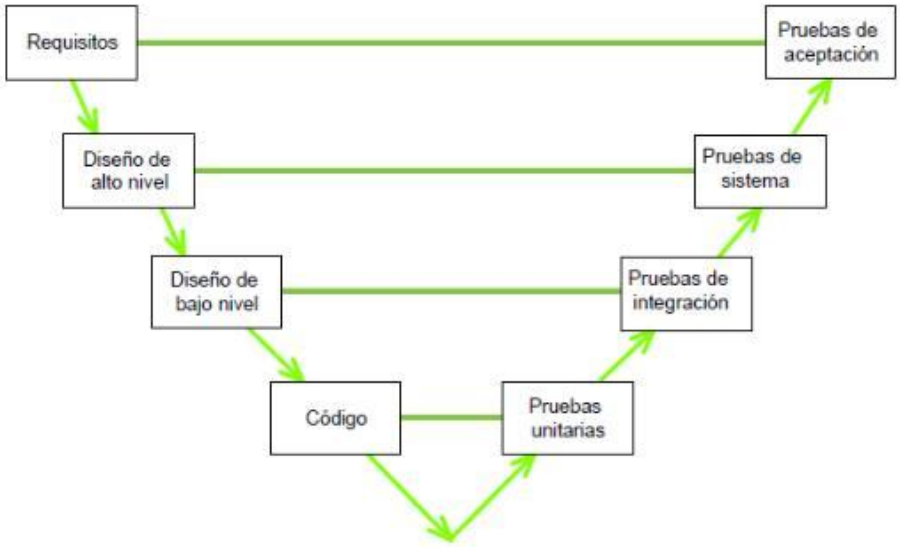


Imagen 34 - Modelo en V

Pruebas a las que se ha sometido al sistema:

- Pruebas unitarias: Es la prueba de cada módulo, que normalmente realiza el propio personal de desarrollo en su entorno. Se prueba el correcto funcionamiento de los módulos del código, asegurando así el perfecto funcionamiento de cada módulo por separado. Se encargan de analizar que un único componente de la aplicación (función o método) devuelva resultados correctos de acuerdo a ciertas entradas.
  Se han realizado pruebas unitarias a todos los formularios, tratando de proseguir el cálculo sin haber completado los campos obligatorios, introduciendo números impares en casillas que requieren números pares, números *float* en campos que solicitan enteros…
  Los módulos que se pueden tratar de forma independiente como: El Control de Calentamiento, El Ajuste de una Máquina, El Control de Reactancia Subtransitoria... también han sufrido pruebas unitarias.

- Pruebas de integración: Aunque previamente se haya demostrado que los módulos funcionan correctamente mediante pruebas unitarias, cabe pensar que el hecho de que varios módulos trabajen de forma conjunta

puede acarrear fallos. Por esa razón realizamos este tipo de pruebas, asegurando que los módulos que están relacionados funcionen correctamente.

Un ejemplo de pruebas de integración que haya realizado al sistema puede ser la Librería para el tratamiento de ficheros. La componen módulos como: crear_einvps, leer_einvps, leer_Ausvps… módulos que se han dado por correctos mediante pruebas unitarias. Para comprobar la correcta integración de la librería en el proceso del *Cálculo Eléctrico* se han realizado pruebas de integración

- Pruebas del Sistema: La fase de pruebas del sistema tiene como objetivo verificar el sistema software para comprobar si este cumple sus requisitos

  Para probar el sistema se han realizado:

  - Pruebas de contenido, verificando que las palabras usadas para transmitir una idea al usuario sean las adecuadas.

  - Pruebas de funcionalidad. Este tipo de pruebas examina si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones de diseño. Para estas pruebas hemos usamos los esquemas de pruebas de caja negra ya que nos interesa saber si funciona o no, independientemente de la forma en que lo haga.

  - Pruebas de usabilidad. Tienen la finalidad de verificar como de fácil de usar el un sistema.

- Pruebas de aceptación: Estas pruebas las realiza el cliente. Son pruebas sobre todo el sistema completo y después de haber realizado todas las pruebas de integración. El cliente comprueba en su propio entorno de explotación si lo acepta como está o no.

El responsable de la aplicación de la empresa INDAR Electric S.L, dio su visto bueno tras un periodo de pruebas en el entorno real, en el que comprobó que el sistema era fiable y cumplía las expectativas.

9. GESTIÓN DEL PROYECTO

Al final de todo proyecto se ha de realizar una contabilidad de los recursos consumidos y su impacto sobre lo inicialmente planificado.

9.1 Parte de horas mensuales:

A continuación, se muestra un desglose mensual de las horas invertidas a lo largo de los meses del proyecto.

NOVIEMBRE

TAREA	TOTAL HORAS
DOP	13,6
ESTUDIO DEL ENTORNO + HERRAMIENTAS	68
CAPTURA REQUISITOS S1	20,4
ANÁLISIS S1	40,8
DISEÑO S1	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	0
CAPTURA DE REQUISITOS S2	0
ANÁLISIS S2	0
DISEÑO S2	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	0
MEMORIA	0
PRESENTACIÓN	0
TOTAL HORAS	**142,8**

Imagen 35 - Horas mensuales Noviembre

DICIEMBRE

TAREA	TOTAL HORAS
DOP	0
ESTUDIO DEL ENTORNO + HERRAMIENTAS	0
CAPTURA REQUISITOS S1	0
ANÁLISIS S1	13,6
DISEÑO S1	88,4
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	0
CAPTURA DE REQUISITOS S2	0
ANÁLISIS S2	0
DISEÑO S2	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	0
MEMORIA	0
PRESENTACIÓN	0
TOTAL HORAS	**102**

Imagen 36 - Horas mensuales Diciembre

ENERO

TAREA	TOTAL HORAS
DOP	0
ESTUDIO DEL ENTORNO + HERRAMIENTAS	0
CAPTURA REQUISITOS S1	0
ANÁLISIS S1	0
DISEÑO S1	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	68
CAPTURA DE REQUISITOS S2	28
ANÁLISIS S2	11,2
DISEÑO S2	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	0
MEMORIA	0
PRESENTACIÓN	0
TOTAL HORAS	**107,2**

Imagen 37 - Horas mensuales Enero

FEBRERO

TAREA	TOTAL HORAS
DOP	0
ESTUDIO DEL ENTORNO + HERRAMIENTAS	0
CAPTURA REQUISITOS S1	0
ANÁLISIS S1	0
DISEÑO S1	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	80
CAPTURA DE REQUISITOS S2	0
ANÁLISIS S2	56
DISEÑO S2	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	0
MEMORIA	0
PRESENTACIÓN	0
TOTAL HORAS	**136**

Imagen 38 - Horas mensuales Febrero

Transcurridos cuatro meses, llegábamos al punto intermedio del proyecto, en el que nos encontrábamos terminando de implementar y probando el Cálculo eléctrico y comenzábamos a dar los primeros pasos en el Generador de planos, analizando y situándonos en el problema, dejando atrás documentos como el DOP.

MARZO

TAREA	TOTAL HORAS
DOP	0
ESTUDIO DEL ENTORNO + HERRAMIENTAS	0
CAPTURA REQUISITOS S1	0
ANÁLISIS S1	0
DISEÑO S1	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	56
CAPTURA DE REQUISITOS S2	0
ANÁLISIS S2	0
DISEÑO S2	76
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	0
MEMORIA	0
PRESENTACIÓN	0
TOTAL HORAS	**132**

Imagen 39 - Horas mensuales Marzo

ABRIL

TAREA	TOTAL HORAS
DOP	0
ESTUDIO DEL ENTORNO + HERRAMIENTAS	0
CAPTURA REQUISITOS S1	0
ANÁLISIS S1	0
DISEÑO S1	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	0
CAPTURA DE REQUISITOS S2	0
ANÁLISIS S2	0
DISEÑO S2	27,2
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	74,8
MEMORIA	0
PRESENTACIÓN	0
TOTAL HORAS	**102**

Imagen 40 - Horas mensuales Abril

MAYO

TAREA	TOTAL HORAS
DOP	0
ESTUDIO DEL ENTORNO + HERRAMIENTAS	0
CAPTURA REQUISITOS S1	0
ANÁLISIS S1	0
DISEÑO S1	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	0
CAPTURA DE REQUISITOS S2	0
ANÁLISIS S2	0
DISEÑO S2	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	129,2
MEMORIA	6,8
PRESENTACIÓN	0
TOTAL HORAS	**136**

Imagen 41 - Horas mensuales Mayo

JUNIO

TAREA	TOTAL HORAS
DOP	0
ESTUDIO DEL ENTORNO + HERRAMIENTAS	0
CAPTURA REQUISITOS S1	0
ANÁLISIS S1	0
DISEÑO S1	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	0
CAPTURA DE REQUISITOS S2	0
ANÁLISIS S2	0
DISEÑO S2	0
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	0
MEMORIA	61,2
PRESENTACIÓN	20,4
TOTAL HORAS	**81,6**

Imagen 42 - Horas mensuales Junio

El mes de Junio, es un mes marcado por la entrega del proyecto. La tarea principal, fue la elaboración de la memoria, debido a que el 18 de Junio la fecha límite para la entrega.

En la siguiente tabla podemos ver el número total de horas empleadas para cada una de las tareas. Como podemos observar, el cómputo total de horas invertidos en el Proyecto Fin de Carrera, asciende a 939,6 horas de trabajo.

TOTAL

TAREA	TOTAL HORAS
DOP	13,6
ESTUDIO DEL ENTORNO + HERRAMIENTAS	68
CAPTURA REQUISITOS S1	20,4
ANÁLISIS S1	54,4
DISEÑO S1	88,4
IMPLEMENTACIÓN + PRUEBAS + CIERRE S1	204
CAPTURA DE REQUISITOS S2	28
ANÁLISIS S2	67,2
DISEÑO S2	103,2
IMPLEMENTACIÓN + PRUEBAS + CIERRE S2	204
MEMORIA	68
PRESENTACIÓN	20,4
TOTAL HORAS	**939,6**

Imagen 43 - Horas Totales

El gráfico 1, muestra las horas empleadas para cada tarea. En el caso del análisis, diseño e implementación + pruebas + cierre, reflejará el cómputo total de las horas empleadas en los dos subproyectos. Destacan las horas empleadas para la implementación.

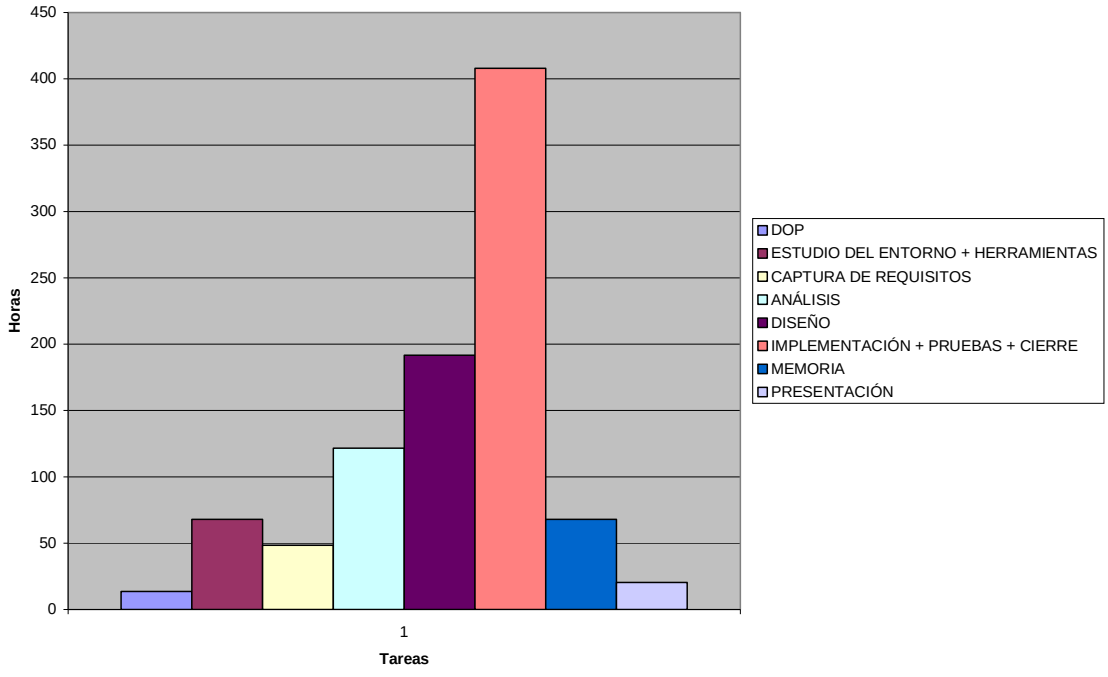


Gráfico 1 – Horas empleadas por tareas

En gráfico 2, se muestra la carga de trabajo que ha supuesto cada uno de las tareas dividida en porcentajes

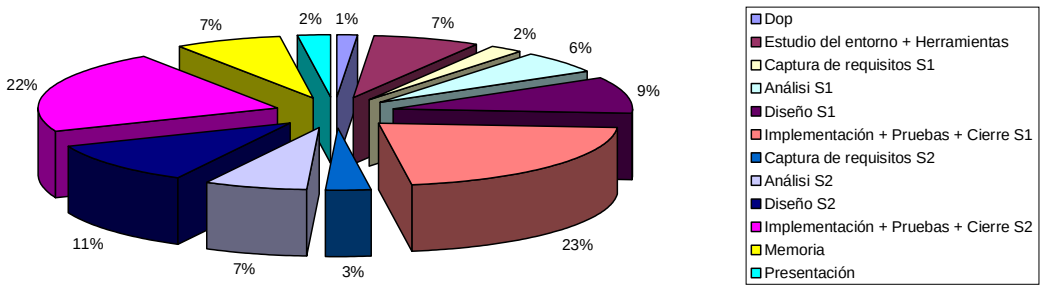


Gráfico 2 – Porcentaje del tiempo empleado por tarea

Horas reales vs Horas planificadas, muestra la comparativa entre las horas planificadas y las reales. Como podemos observar el mayor desfase se ha producido en la estimación del estudio del entorno, en el análisis y en el diseño.

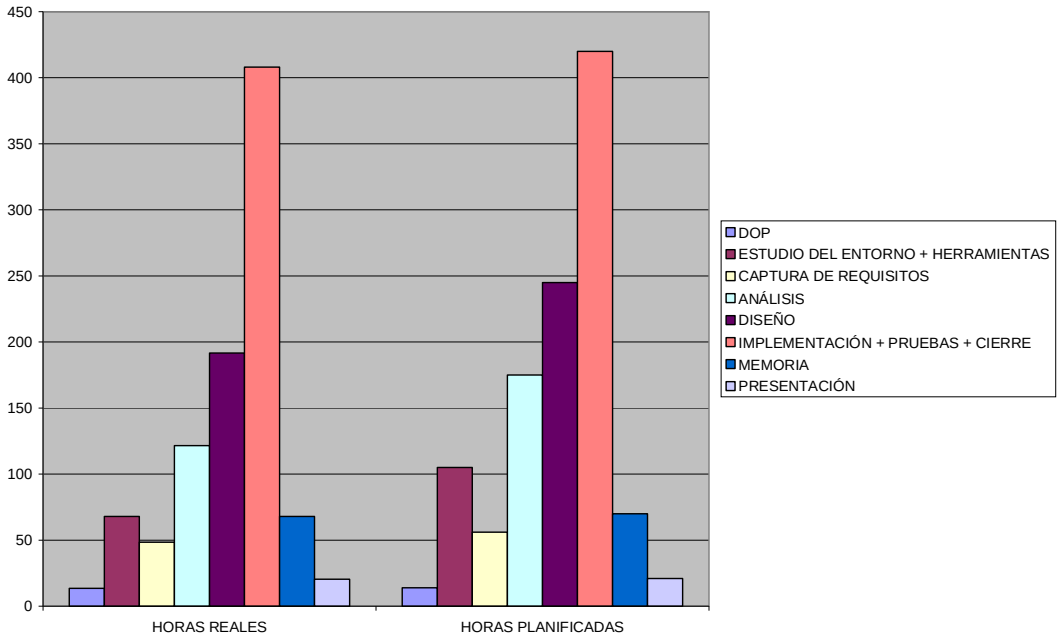
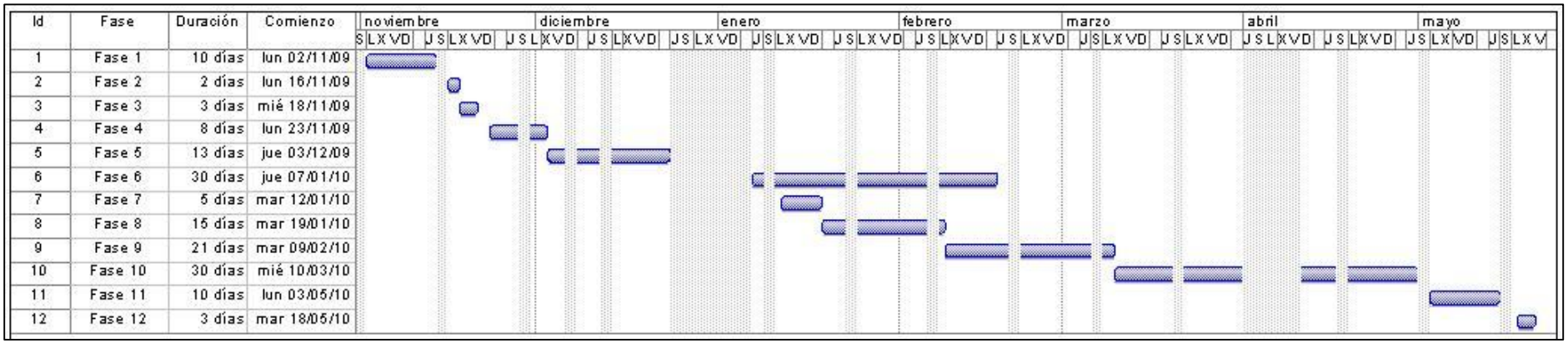


Gráfico 3 - Horas reales vs Horas planificadas

9.2 Gantt planificado vs real:

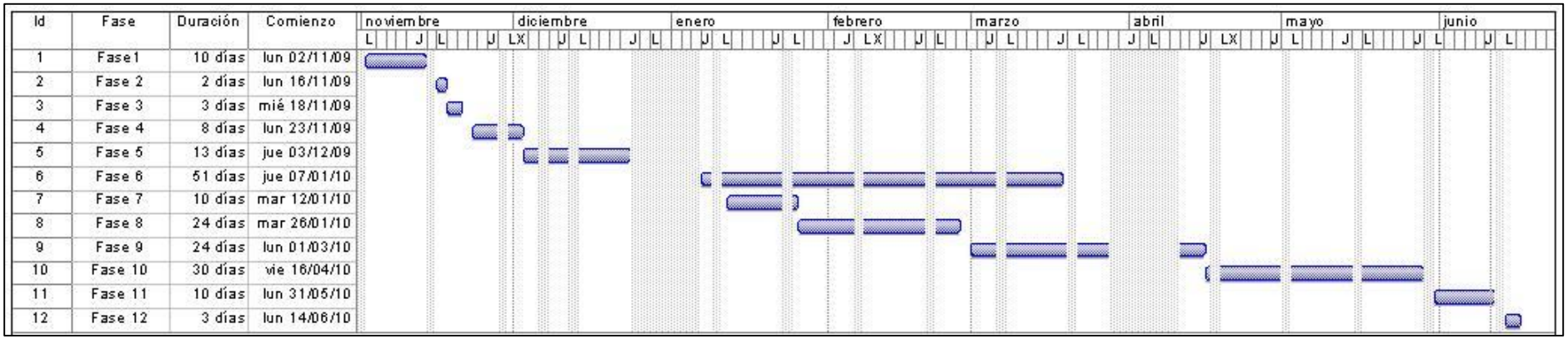
Planificación inicial:



Id	Fase	Duración	Comienzo
1	Fase 1	10 días	lun 02/11/09
2	Fase 2	2 días	lun 16/11/09
3	Fase 3	3 días	mié 18/11/09
4	Fase 4	8 días	lun 23/11/09
5	Fase 5	13 días	jue 03/12/09
6	Fase 6	30 días	jue 07/01/10
7	Fase 7	5 días	mar 12/01/10
8	Fase 8	15 días	mar 19/01/10
9	Fase 9	21 días	mar 09/02/10
10	Fase 10	30 días	mié 10/03/10
11	Fase 11	10 días	lun 03/05/10
12	Fase 12	3 días	mar 18/05/10

Imagen 44 - Gantt planificado

Planificación real:



Id	Fase	Duración	Comienzo
1	Fase1	10 días	lun 02/11/09
2	Fase 2	2 días	lun 16/11/09
3	Fase 3	3 días	mié 18/11/09
4	Fase 4	8 días	lun 23/11/09
5	Fase 5	13 días	jue 03/12/09
6	Fase 6	51 días	jue 07/01/10
7	Fase 7	10 días	mar 12/01/10
8	Fase 8	24 días	mar 26/01/10
9	Fase 9	24 días	lun 01/03/10
10	Fase 10	30 días	vie 16/04/10
11	Fase 11	10 días	lun 31/05/10
12	Fase 12	3 días	lun 14/06/10

Imagen 45 - Gantt real

9.3 Conclusiones de la gestión:

En todo proyecto, como en la vida, surgen imprevistos que te obligan a aumentar o a prolongar la dedicación inicial planeada, retrasando otras tareas o propósitos. Un proyecto sin imprevistos, es una utopía. No es real pensar que no vamos a tener ninguna dificultad o impedimento. Por ello, es de gran importancia hacer una planificación inicial lo mas real y objetiva posible.

En el caso de los Proyectos Fin de Carrera, este tipo de gestiones, las realiza gente inexperta, como somos los universitarios. Es natural observar al final del proyecto, que la estimación de esfuerzo inicial, era exageradamente corta o larga. En nuestro caso, nuestras estimaciones en horas, no difieren tanto la planificada de la real. La diferencia principal radica en la planificación diaria a la hora de empezar a trabajar con el segundo subproyecto paralelamente, en ese periodo nos dimos cuenta de que las tareas se iban a empezar a retrasar debido a que nuestra dedicación diaria se repartía en los dos subproyectos, por lo que no necesitaba más horas, pero si mas días para realizar la tarea.

A continuación expongo los temas a los que hemos dedicado más tiempo, imprevistos que nos han surgido o en el peor de los casos, problemas que nos han obligado a retrasar o modificar la planificación establecida a medida que ha ido avanzando el proyecto.

Una lección importante, cuando te enfrentas a un trabajo de este calibre, no dependes únicamente de ti mismo. Tienes que trabajar con mucha gente, preguntar muchas cosas, pedir información, pedir que te realicen determinadas tareas que quedan fuera de tu campo y que necesitas para proseguir con tus objetivos, que te proporcionen recursos, consejos e incluso, que te brinden ayuda en muchos ocasiones. Todo esto y mucho más exige una planificación previa y conjunta para poder llevar adelante el proyecto y a tiempo. En mi opinión, este es el factor mas imprevisible y muy a tener en cuenta a la hora de realizar una buena gestión de proyecto.

Reingeniería: Como he mencionado anteriormente, la empresa contaba con código que realizaba el cálculo eléctrico. Dicho código, estaba programado línea a línea, no era una programación modular ni estructurada y uno de los objetivos era corregir tanto aspectos técnicos, como la programación. Debido a esto, una de nuestras primeras tareas fue enfrentarnos a un código del que no había documentación asociada. Código desestructurado, programado por otra persona, implementado en un lenguaje y en un entorno desconocido. De este código debía extraer cómo se hacían los cálculos, que proceso seguía, y volverlo a implementar. Esta tarea, me llevó más esfuerzo de lo planificado.

Del punto anterior se aprende la importancia de añadir una buena documentación a un código; de cara a futuras mejoras, modificaciones o para reutilizar módulos existentes. Para facilitar dicha labor, se han redactado varios documentos por cada subproyecto (Cálculo eléctrico y Generador de Planos) y la librería para el tratamiento de ficheros einvps y Ausvps. No es una tarea complicada pero es una dedicación que no había previsto al inicio del proyecto.

Para cada subproyecto se han redactado los siguientes documentos:

1. Arquitectura: El siguiente documento pretende reflejar la arquitectura de funcionamiento y lógica seguidos para el desarrollo del software.

2. Lógica de negocio: En este documento se muestra cada clase y su estructura, todos los atributos, el tipo y ámbito de cada uno de ellos. A continuación se explica la función de cada método mediante pseudocódigo, especificando cuales son sus parámetros de entrada y cuales los de salida.

3. Tratamiento de datos: En este otro documento, se mostrará en caso de haber utilizado una base de datos, que tablas se han utilizando, una pequeña definición de cada tabla así como los datos que la componen.

Para la librería:

1. Lógica de negocio: En este documento se muestra cada clase y su estructura, todos los atributos, el tipo y ámbito de cada uno de ellos. A continuación se explica la función de cada método mediante pseudocódigo, especificando cuales son sus parámetros de entrada y cuales los de salida.

Como he mencionado en otras ocasiones, para tratar los datos de los ficheros que debía manejar, optamos por crear un módulo independiente que se encargase de esas operaciones en concreto. Un módulo independiente y reutilizable, una librería. Al tratarse de cientos de atributos, buscamos y hallamos después de un tiempo considerable, herramientas como *Reflection*, que nos facilitaban la labor de implementación. El desarrollo de esta librería y de su documentación asociada, ha supuesto muchas horas de trabajo.

Uno de los puntos de más carga de trabajo, fue, lanzar y conseguir manejar AutoCAD 2010 desde VB.net. Para lograrlo, tuvimos que consultar libros, manuales y visitar multitud de foros, en este último recurso fue donde encontramos la mayor ayuda. Muchas soluciones a problemas que se me antojaban, tenían su respuesta en alguna de las entradas de foros como HISPACAD. La búsqueda de todas las herramientas para utilizar AutoCAD desde VB.net supuso muchas horas de trabajo.

Un aspecto que tiene sus ventajas y desventajas, es la de haber realizado un proyecto real y en empresa. Hablando desde el punto de vista de la gestión, ha sido un trabajo organizado y con un seguimiento continuo. Es decir, mi objetivo era llegar a la convocatoria de Junio, pero la empresa por su parte, también ha establecido objetivos y fechas que había que cumplir.

- Ventajas: Te obliga a coger una dinámica de trabajo diario y a cumplir tu planificación.

- Desventajas: Trabajar cuando tienes una fecha encima, no encuentras las herramientas necesarias para satisfacer los objetivos de la empresa provoca momentos de agobio y estrés, sensaciones que se convierten en plena satisfacción cuando logras aquello que tanto te ha costado y has conseguido completar exactamente como querías o te pedían.

10. CONCLUSIONES

Llegados a este punto, es conocido que se han cumplido los objetivos del proyecto, que se ha finalizado con éxito y en los tiempos esperados.

El Cálculo eléctrico es un ejecutable capaz de efectuar el proceso de cálculo para la obtención del generador hidráulico óptimo del catalogo de ofertas INDAR, según ciertas características de entrada definidas por el cliente. La aplicación está integrada tanto en la Intranet como en la Extranet de la empresa INDAR Electric S.L.

Generación de Planos es otro ejecutable capaz de manejar información contenida en un fichero generado de forma automática por el Configurador de Ofertas (consultar Anexo 1) en el caso de que la aplicación Generador de Planos, se lance como extensión del Configurador de Ofertas, por el contrario, si dicha aplicación es lanzada de forma independiente, el ejecutable será capaz de extraer la información introducida por el cliente en un formulario. Mediante el uso de una librería de imágenes, genera planos de oferta de generadores hidráulicos INDAR, con cotas y textos.

Las aplicaciones han completado la fase de prueba y están integradas tanto en la Intranet como en la Extranet de la empresa, por lo que podemos decir que son una realidad. A día de hoy, se resuelven ofertas de clientes utilizando tanto El Cálculo Eléctrico como el Generador de planos.

Se estima que mediante el Configurador de Ofertas para Generadores Hidráulicos, aplicación general que integra los módulos Cálculo Eléctrico y Generador de planos, se puede satisfacer cerca del 80% de las peticiones de oferta recibidas por la línea de hidráulica INDAR Electric S.L. La razón por la que el 20% restante queda fuera de su alcance, se debe a que las condiciones del generador solicitado no entran en los parámetros considerados como estándar o habitual.

Este proceso de respuesta que antes podía requerir en el mejor de los casos cerca de 2 horas, mediante el Configurador de Ofertas se puede realizar en unos 15-20min dependiendo de la complejidad de cálculos y exigencias del cliente.

Mediante el Cálculo Eléctrico y del Generador de Planos, principalmente se pretende reducir el tiempo de respuesta y el gasto de recursos humanos a la hora de afrontar la resolución de una oferta, con la sola necesidad de una persona (Dep. Comercial).

Otro aspecto muy importante es que las aplicaciones son ejecutables desde cualquier lugar del mundo y de forma concurrente, posibilitando de este modo, poder dar respuesta a una petición de oferta desde cualquier sitio con conexión a Internet a cualquier hora, sin depender de ningún otro factor.

El preservar la confidencialidad de la empresa, nos ha obligado a omitir mucho volumen de información y ha dificultado poder explicar con claridad: la estructura de las clases, los archivos que manejo, la implementación, los datos de entrada de los formularios, o poder mostrar con claridad y con ejemplos, cual es el resultado de cualquiera de las dos aplicaciones.

En lo relativo a lo personal, haber llevado acabo proyectos simulados en asignaturas como Ingeniería del Software, ayuda mucho a la hora de organizarse al inicio del proyecto, pero no dejan de ser ejercicios de clase que pretenden acercarse a la vida laboral. En el transcurso de estos 8 meses, he tenido que organizar y asistir a multitud de reuniones redactando posteriormente sus actas correspondientes, en las que se planteaban acciones a desarrollar para fechas concretas y por distintos integrantes del grupo. Planificar y gestionar las tareas para una óptima organización, es la acción más difícil que se presenta en este tipo de trabajos. Sin una buena planificación, es probable que alguno de los trabajadores se quede pendiente del trabajo de los demás sin poder avanzar en sus propias tareas, no se cumplan los plazos establecidos, además de por los imprevistos surgidos, por la mala planificación temporal… Nunca es plato de buen gusto asumir errores propios, aunque sean

en áreas en los que todavía se es inexperto. Pero esta experiencia que he adquirido en estos 8 meses, forma parte del aprendizaje que aplicaré en futuros proyectos.

Un mundo como el informático requiere mucha inquietud y de un proceso de aprendizaje continuo, es indispensable seguir aprendiendo. Algunas de las herramientas aquí utilizadas, eran completamente desconocidas para mí, y aunque no puedo afirmar poseer un dominio absoluto a día de hoy, soy capaz de manejarme con soltura en cualquiera de las herramientas que he utilizado.

11. ANEXO I

¿Qué es el Configurador de Ofertas?

Anteriormente a la existencia del Configurador de Ofertas para Generadores Hidráulicos, la línea de hidráulica de INDAR Electric S.L no disponía de una herramienta que por si misma fuese capaz de responder por completo a una petición de oferta.

El proceso que aplicaban consistía en la recepción de la petición de oferta por parte del Dep. Comercial. Este departamento, mediante el uso de unas plantillas estándar para todas las líneas, facilitaba los parámetros que debería cumplir el generador a los calculistas eléctricos. Estos, ajustándose a dichos parámetros. Efectuaban los cálculos y ajustes necesarios hasta llegar a definir el tipo de generador válido para satisfacer la petición de oferta en lo que respecta exclusivamente eléctrico; empleando para ellos una aplicación de cálculos eléctricos desarrollado en exclusiva para INDAR Electric S.L. tras definir la máquina eléctricamente, el siguiente paso consistía en el cálculo mecánico. Los calculistas mecánicos recibían toda la información existente hasta el momento (requerimientos del cliente y resultados del cálculo eléctrico) y basándose en ellos, definían el generador mecánicamente (principalmente pesos, diseño y dimensiones) realizando diferentes cálculos mecánicos; llevando a cabo algunos de ellos mediante diferentes hojas de cálculo Excel desarrollados en INDAR Electric S.L. Una vez realizada la definición mecánica, los propios calculistas mecánicos, calculaban costes eléctricos y mecánicos. Tras todos estos cálculos, la oferta con todos los resultados era remitida al Dep. Comercial. El siguiente cometido del comercial responsable de la oferta era definir todos los accesorios y auxiliares que complementarían el generador. De mismo modo, establecía los servicios, puntos de inspección y condiciones comerciales referentes al generador de los que dispondría la empresa. Para concluir estableciendo el precio final y redactando el informe final de la oferta.

El Configurador de Ofertas es una aplicación que engloba la automatización de los cálculos anteriormente definidos.

12. BIBLIOGRAFÍA

Libros:

- AutoCAD 2006 VBA: a programmer's reference
  Autor: Joe Sutphin
  Editorial: Apress, 2005

- Introducción a ASP.NET
  Autor: Consultec, S.L

- Ingeniería del software: Un enfoque práctico
  Autor: Roger S. Presuman

- Apuntes de la asignatura Ingeniería del Software
  Autor: German Rigau

Paginas Web:

- Url: http://www3.uji.es/~jperis/dfao/apuntes/vba.pdf
  Descripción: Resumen de comandos VBA para AutoCAD

- Url: http://www.google.es/
  Descripción: Buscador

- Url: http://www.hispacad.com/
  Descripción: Foro de dudas de VB/VB.NET para AutoCAD

- Url: http://es.wikipedia.org/wiki/Wikipedia:Portada
  Descripción: Enciclopedia de contenido libre

- Url: http://www.elguille.info/
  Descripción: Web de información y foros para los programadores

- Url: http://msdn.microsoft.com/es-es/default.aspx
  Descripción: API o biblioteca MSDN

- Url: http://forums.autodesk.com/t5/AutoCAD-2010/bd-p/360
  Descripción: Foro de AutoDesk en Ingles