Universitat Autònoma
de Barcelona

ESCRITORIO DEL GESTOR

Memoria del proyecto de
Ingeniería Técnica en
Informática de Sistemas
realizado por

Pablo Máximo Elorga Martin

y dirigido por

Jordi Pons Aróztegui

Escola d'Enginyeria

Sabadell, julio de 2011

Jordi Pons Aróztegui, profesor de la Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su dirección por **Pablo Máximo Elorga Martín**

Y para que conste firma la presente.
Sabadell, **Julio** de **2011**

Firmado: **Jordi Pons Aróztegui**

Jordi Soler Beteta, de la empresa UNIT4,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria ha sido realizado bajo su tutorización por **Pablo Máximo Elorga Martin**

Y para que conste firma la presente.
Sabadell, **julio** de **2011**

Firmado: **Jordi Soler Beteta**

HOJA RESUMEN – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Titulo del proyecto: Escritorio del Gestor	
Autor: Pablo Máximo Elorga Martin	**Data:** *Julio 2011*
Tutores: Jordi Pons Aróztegui (UAB) / Jordi Soler Beteta (UNIT4)	
Titulación: Ingeniería Técnica de Informática de Sistemas	

Palabras clave

- Castellano: escritorio, gestor, multiaplicación.
- Catalán: escriptori, gestor, multiaplicació.
- Inglés: desktop, manager, multitask.

Resumen del proyecto

- Castellano:
  El proyecto tiene por finalidad crear una aplicación multifunción que permita a un gestor visualizar la información más relevante e importante respecto al conjunto de colaboradores que tiene a su cargo y que puede ser útil en la toma de decisiones a corto y largo plazo. Dicho proyecto se ha desarrollado en la delegación de UNIT4 Ibérica de Barberà del Vallès.

- Catalán:
  El projecte té per finalitat crear una aplicació multifunció que permeti a un gestor visualitzar la informació més rellevant i important respecte al conjunt de col·laboradors que té al seu càrrec i que pot ser útil en la presa de decisions a curt i llarg termini. Aquest projecte s'ha desenvolupat a la delegació de UNIT4 Ibèrica de Barberà del Vallès.

- Inglés:
  The project aims to create a multifunction application that allows a manager to visualize the most relevant and important for all of the employees who have to responsible that can be useful in making short and long term. The project has been developed in the Iberian UNIT4 delegation Barberà del Vallés.

Tabla de Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

1.1. Marco de Trabajo

1.1.1. Convenio de colaboración entre UNIT4 y la UAB

Este proyecto se ha desarrollado dentro del marco de un convenio entre la Universitat Autònoma de Barcelona y la empresa UNIT4, en el cual, se estipulan 560 horas para realizar un trabajo de desarrollo de software. Mediante este convenio, el alumno puede adquirir conocimientos de cómo funciona una empresa de desarrollo de software de gestión y de tecnologías de la información, a la vez que este desarrollo sirve como trabajo de final de carrera.

1.1.2. La empresa

UNIT4 tiene una experiencia durante más de cuarenta años en el mundo de la ingeniería de software. Posee un amplio abanico de soluciones de gestión de última generación lo cual la hace ser una de las primeras empresas españolas en tecnologías de la información y comunicaciones.

UNIT4 cuenta con 4.230 empleados entre empleados y distribuidores repartidos a lo largo de todo el mundo para garantizar un acceso fácil y local a las ventas, servicios y soporte. Se encuentran en: Alemania, Australia, Bélgica, Canadá, Dinamarca, España, Estados Unidos, Estonia, Francia, Holanda, Hungría, Irlanda, Malasia, Noruega, Portugal, Reino Unido, República Checa, Singapur, Suráfrica, Suecia y Uganda.

Mediante la oferta de soluciones y objetivos, su misión es la de proporcionar una auténtica ventaja competitiva y diferenciadora a sus clientes mediante la implantación de soluciones informáticas de gestión.

Sus principales objetivos como empresa son:

- Mantener su posición de liderazgo como proveedor global de soluciones y servicios TIC.
- Estar por delante en investigación de nuevas tecnologías y desarrollo de nuevas soluciones.

Por los requisitos del proyecto éste se sitúa dentro del departamento de *“ekon Ágora”* que es el grupo que se encarga del desarrollo y el soporte de la herramienta del producto Ágora basado en la tecnología karat, desarrollando nuevas aplicaciones y manteniendo las ya existentes.

1.1.3. Karat ekon y walnut.

Karat: es la plataforma tecnológica de UNIT4 para el desarrollo, implantación, mantenimiento y evolución continua del software de gestión de la empresa. Así mismo es la herramienta que permite la creación de los diferentes productos de la empresa UNIT4, ofreciendo una amplia variedad de opciones a la hora de crear nuevas soluciones tecnológicas.

ekon: es el software de gestión de empresa compuesto de diferentes de productos UNIT4 basados en tecnología karat diseñado para adaptarse a las necesidades específicas de cada una de las empresas, sin renunciar a las ventajas del software estándar y con una visión de total integración de procesos.

Walnut: son una serie de directrices de diseño, codificación e integración que buscan la mejora de la usabilidad de las aplicaciones y el aprendizaje por parte del usuario final de la aplicación para mejorar la experiencia y conocimiento de las mismas.

1.2 Objetivos del proyecto

A nivel de aplicación tecnológica se busca desarrollar una aplicación que permita la visualización global de la información más relevante y que tenga una incidencia directa en la toma de decisiones de una determinada empresa además de adaptar todas las funcionalidades para que sean sencillas y útiles. Ya que los usuarios no se vean desbordados por las posibilidades de la herramienta.
Así mismo debe incorporar todas las herramientas necesarias para que el usuario tenga una experiencia satisfactoria a la hora de interactuar con la aplicación.

A nivel personal, como desarrollador y programador, se busca adquirir experiencia en el trabajo diario dentro de una gran empresa y aprender metodologías de trabajo, aprender Java, un lenguaje de programación orientado a objetos con un gran uso en el mundo laboral y por último desarrollar una aplicación a partir de una aplicación base que sirve como punto de partida de la propia aplicación creada, siguiendo todas las fases de desarrollo de software (investigación, análisis, codificación y pruebas).

1.3. Agradecimientos

Agradecer a la Universidad Autónoma de Barcelona la posibilidad de realizar el proyecto en una empresa de ámbito tecnológico puntera en su sector como es UNIT4.
A UNIT4 por brindarme la oportunidad de realizar mi Proyecto de fin de carrera en sus instalaciones aportándome las ayudas necesarias en el ámbito académico, material y soporte informático para desarrollarlo además de vivir la experiencia de trabajar en entorno de trabajo en el desarrollo de software.

A mis tutores, tanto de la UAB como de UNIT4 y a las personas delegadas de ellos que han permitido la creación de este proyecto así como las personas de otros departamentos que sin dudarlo han aportado una gran parte en el desarrollo de la aplicación final.

Por último a mis compañeros de trabajo y a la vez de universidad que aunque todos realizamos proyectos distintos, han aportado conocimientos y diversos puntos de vista que han ayudado a la creación de este Proyecto.

1.4. Contenido de la memoria

Seguidamente, se describen los puntos de que consta la memoria del proyecto:
· **Introducción**: Apartado en el cual se explica de forma breve el marco de ubicación del proyecto, así como los objetivos marcados.
· **Definición de requerimientos**: Se hace una pequeña descripción del sistema actual, del sistema propuesto (objetos, funcionalidades, etc.) y de los beneficios y riesgos del proyecto.
· **Planificación del proyecto**: En este capítulo se puede observar la planificación inicial de las tareas a realizar en el proyecto con su duración, y su desviación final. También se explica el método de desarrollo de software utilizado.
· **Implementación**: Se profundiza en lo especificado en el capítulo anterior y se especifican sus componentes en los dos niveles, así como en la descripción de los procesos involucrados. También se realiza un plan de pruebas a realizar una vez codificada la aplicación.
· **Construcción**: Apartado en el cual se describe la tecnología utilizada en el desarrollo, así como el estilo de codificación usado.
· **Pruebas**: Se describen en este apartado todas las pruebas realizadas sobre la aplicación y el resultado de éstas para comprobar la eficacia del programa.
· **Conclusiones**. Se enumeran los objetivos conseguidos, las líneas de trabajo abiertas así como una valoración personal sobre el proyecto.
· **Bibliografía**: Detalle de recursos informativos utilizados para realizar el aplicativo.

MARCO DE DESARROLLO: ÁGORA

2.1. Introducción

Ágora es una avanzada solución desarrollada por UNIT4, con tecnología karat, que permite la automatización de procesos internos ajenos al negocio principal de la compañía.

Es un producto que permite planificar, gestionar los horarios, controlar la presencia de los Colaboradores de la empresa, controlar las notas de gastos, así como gestionar las peticiones de viajes para la ayuda en la toma de decisiones.

Las causas por las cuales se ha creado Ágora es porque actualmente empresas tienen necesidades administrativas que al no estar informatizadas suponen un gasto importante. ekon ágora aporta la solución a través de una suite de productos de apoyo a las gestiones administrativas, lo que permite una mayor eficiencia en la toma de decisiones y planificación de eventos o tareas

Actualmente consta de los siguientes módulos:

- Planificación de horarios y tareas
- Control de presencia
- Gestión de notas de gastos
- Gestión de viajes

2.2. Descripción de módulos

Planificación de horarios y tareas:
Módulo destinado a la planificación de los horarios de los colaboradores de una empresa. En este módulo se planifica el día a día de un determinado colaborador. Permite gestionar la planificación diaria (jornada), vacaciones, días libres, horas libres, peticiones, etc.

Control de presencia:
Módulo destinado al control horario de los colaboradores y control de tareas. Se encarga del registro de marcas de entrada / salida, pausa, tareas y eventos.
Gestiona la presencia, absentismo, horas extras, puntualidad, visitas, parking, etc. parking, etc. y es el encargado del circuito de petición y validación de permisos y vacaciones.

Gestión de notas de gastos

Se definen los conceptos, plantillas de gasto y formas de abono.

Además este módulo permite registrar, validar, imprimir y entregar comprobantes referentes a los gastos por parte del colaborador

Gestión de viajes

Módulo que gestiona la petición de viajes que los colaboradores llevan a cabo en el desarrollo de su trabajo

Estos son los cuatro grandes pilares que forman el producto Ágora. Dichos módulos contribuyen a que el total de los colaboradores estén claramente definidos con sus respectivos parámetros permitiendo de esta forma interacciones con otros módulos de ekon.

Ekon permite que los diferentes parámetros o datos que contienen cada uno de sus módulos sean reutilizados por otros módulos dando a la empresa una avanzada herramienta de gestión. Ágora es un complemento para ekon Laboro, el modulo encargado de la creación de nominas y los diferentes cómputos de un colaborador entre otras funcionalidades

En este caso, el Escritorio del Gestor se basa en el modulo de planificación y control de presencias. Más adelante se detalla el porqué de esta elección.

2.1. Contexto

Actualmente el producto Ágora hace uso de diferentes aplicaciones para mostrar el conjunto de la información referente a un determinado grupo de colaboradores; para modificar, confirmar o eliminar dicha información el usuario debe interactuar con un conjunto de pantallas que se despliegan independientemente unas de otras.

Dicha información se muestra de forma independiente y no se muestra relación aparente entre ellas, por lo que el usuario debe abrir diferentes pantallas para una visualización completa de la información.

Estas aplicaciones aunque guardan una relación de base entre ellas (utilizan los mismos datos) la información se muestra de una forma separada y sin la posibilidad de que el usuario pueda tener delante los datos anteriormente consultado, obteniendo una vista en cascada de las diferentes ventanas.

Las pantallas de Ágora permiten la consulta, modificación o eliminación de determinados eventos, incidencias o peticiones por parte de diferentes colaboradores que un gestor tiene bajo su responsabilidad.

Estas pantallas están compuestas por diferentes formularios[1] donde se representa el total de la información. Mediante los formularios el gestor puede visualizar toda la información referente al conjunto de colaboradores y filtrar dicha información según sus necesidades. Otro conjunto de pantallas permite la búsqueda de colaboradores o personal a su cargo, el marcaje y control de fichajes a si como el detalle de los mismos.

Cada pantalla cumple una determinada función dentro del módulo Ágora siendo complementarias en algunos casos. Estas pantallas, aunque por separado, muestran toda la información necesaria para el gestor de los colaboradores.

Algunas de las pantallas más relevantes de Ágora son las siguientes:

- Registro de marcas: permite la realización de marcas de forma manual. Se observa en detalle el total de los fichajes. Por otra parte se muestran diferentes acciones como peticiones, consultas modificaciones y pausas.
- Modificar marcas diarias: permite seleccionar un día y podemos observar las marcas realizadas. A la vez nos permite modificar el tipo, la hora, etc. en caso de ser errónea podemos modificarla indicando un motivo.
- Registro de presencia por colaborador: muestra las diferentes jornadas realizadas y si ha habido alguna incidencia.
- Consulta de planificación anual: visualizamos la planificación de un determinado colaborador a lo largo de un año o mes seleccionado con todas las incidencias y peticiones ya incluidas.

2.2. Diagnóstico

La forma actual de interactuar con Ágora es poco intuitiva y se debe conocer la funcionalidad de cada una de las pantallas que permiten su interacción. Así mismo se debe conocer que funcionalidad nos permite realizar cada una de ellas y los cambios que pueden provocar.

El usuario debe leer primero el manual de funcionamiento de la aplicación para poder saber cómo interactuar con ella, no permite que un usuario de un principio vea el funcionamiento básico de la aplicación, cosa que dificulta mucho su uso e interacción.

La aplicación tiene un conjunto de menús o árbol de navegación que impide que un usuario pueda ver varias ventanas a la vez cosa que implica la visión de la información en conjunto. Además la información requiere de la interacción del usuario, es decir, la información requiere de un paso previo por parte del usuario para poder visualizar los datos.

[1] Formulario es un objeto en el cual se muestra un conjunto de registros que contienen cierta información relevante con los cuales el usuario puede interactuar de forma directa para su modificación o visualización.

No obstante el sistema actual cumple con las funcionalidades para la gestión de un conjunto de colaboradores. Lo que se debe mejorar es su visibilidad y facilitar su manejo. Por ello de propone la creación de una nueva aplicación que englobe los aspectos más importantes e incorporarla en una aplicación que mejore dichos aspectos.

ESTUDIO DE VIABILIDAD

3.1. INTRODUCCIÓN

3.1.2. Descripción

Unit4 dispone de *karat*, tal como antes se ha comentado, una completa plataforma tecnológica para la gestión de las empresas, que aporta un nuevo concepto de soluciones basado en la independencia total y real de entornos.

Se trata de la nueva interfaz de interacción con el usuario de la nueva generación de las soluciones *ekon* desarrolladas por Unit4. Desarrollada en Java, es independiente del entorno TI del cliente (Linux, Unix, Apple, Windows, etc.), no precisa de explorador para trabajar en web, puede usarse en cualquier dispositivo (PC, PDA, teléfonos móviles, etc.) y soporta todo tipo de idiomas y alfabetos (Unicode).

Karat ha incorporado una herramienta que permite la incorporación de múltiples pantallas de un producto ekon en una sola pantalla, la que permite al usuario una visualización global de la información desde diferentes ventanas. Es totalmente personalizable, el usuario puede añadir tantas ventanas como espacio disponga en la ventana general además de su libre movimiento. De esta forma karat añade una evolución en la visualización y visión de la información, dando al usuario una total libertad de uso que antes no estaba contemplada. Por lo tanto se deben detectar los aspectos que dificultan la interacción del usuario con el sistema y detallar la finalidad del nuevo sistema a desarrollar.

Finalidad o justificación del sistema a construir:
Se desea evitar que el gestor o usuario de la aplicación para visualizar la información, efectuar cambios, validar peticiones de sus colaboradores, etc. en definitiva gestionar a los colaboradores tenga que ir abriendo diferentes ventanas que se superponen una delante de la otra al ir accediendo a los diferentes menús. Además dichas ventanas muestran la información de forma conjunta y en bloque lo que dificulta su lectura y pudiendo ocasionar que el usuario obvie algún dato relevante ya que toda la información tiene el mismo aspecto.

Actualmente la visualización de la información por separado, no es conjunto lo cual resulta un tanto engorroso recordando datos innecesarios además de ser una aplicación poco intuitiva, puesto que un usuario sin experiencia en la interacción la aplicación puede tener serios problemas a la hora de determinar el tipo de opciones que desea visualizar.

Se propone un escritorio gestor en el que toda la información relevante sea mostrada de forma directa y con la mínima interacción con la aplicación. El gestor es capaz de

gestionar todos los datos importantes a la vez que consulta los datos referentes a ese colaborador o al conjunto de colaboradores.
El escritorio debe ahorrar tiempo y esfuerzo al gestor proporcionando una mayor productividad en su gestión. Además las funcionalidades están diferenciadas puesto que cada ventana realiza una función distinta a las demás.

3.1.3. Objetivos del proyecto

A continuación detallamos los objeticos propuestos para el proyecto así como su clasificación según su importancia en tres categorías: críticos, prioritarios y secundarios.

1. Integrar en una ventana el control de presencia de los colaboradores.
2. Integrar en una ventana la planificación de colaboradores.
3. Integrar en una ventana las peticiones de los colaboradores.
4. Integrar en una ventana las incidencias de los colaboradores.
5. Integrar en una ventana las peticiones de los colaboradores.
6. Integrar en una ventana los límites[2] de los colaboradores.
7. Integrar en una ventana los marcajes de los colaboradores.
8. Posibilidad de aceptar o rechazar peticiones.
9. Posibilidad de aceptar o rechazar incidencias.
10. Permitir el fichaje de los colaboradores y del propio gestor.
11. Añadir más funcionalidades a las respectivas ventanas.

	Crítico	Prioritario	Secundario
O1	×		
O2	×		
O3	×		
O4	×		
O5	×		
O6	×		
O7	×		
O8		×	
O9		×	
O10		×	
O11			×

Tabla 1: Tabla de clasificación de objetivos.

[2] Límite: concepto que describe el tiempo de disfrute de días libres, vacaciones, horas y días de petición

3.1.4. Partes interesadas

A continuación presentemos las partes interesadas del proyecto.

Nombre	Descripción	Responsabilidad
Dept. ekon Ágora UNIT4	Departamento de la empresa UNIT4 dedicado a la parte de desarrollo y mantenimiento del producto ekon Ágora.	Departamento encargado de desarrollar y mantener el producto ekon Ágora

Tabla 2: Tabla de las partes interesadas.

3.1.5. Equipo de proyecto

En la figura siguiente se muestra como queda confeccionado el equipo del proyecto y una pequeña explicación de la responsabilidad de cada uno.

Nombre	Descripción	Responsabilidad
Jordi Pons Aróztegui	Tutor del Proyecto(TP)	Supervisa el trabajo del alumno durante el proyecto.
Jordi Soler Beteta	Jefe de Proyecto UNIT4 (CP)	Define, gestiona y controla el proyecto.
Pablo Máximo Elorga Martín	Analista (A)	Desarrolla el estudio de viabilidad y la planificación. Análisis de la aplicación: arquitectura, metodología, especificación, estándares... Participa en el diseño y validación.
Maite Bastús Lluis	Analista (A)	Asesoramiento en el análisis de la aplicación y asesoramiento técnico a lo largo del proyecto.
Pablo Máximo Elorga Martín	Programador (P)	Desarrolla la aplicación de acuerdo al análisis y la planificación prevista. Hace la implantación del proyecto.
Pablo Máximo Elorga Martín	Diseñador (D)	Diseña y desarrolla el aspecto de la aplicación de acuerdo al análisis y normas de usabilidad establecidas.

Tabla 3: Tabla del equipo del proyecto.

SITUACIÓN ACTUAL

4.1. ESTUDIO DE LA SITUACIÓN ACTUAL

4.1.1. Contexto

Actualmente en el producto ekon Ágora se dispone de un conjunto de formularios que representan el total de la aplicación. Estos formularios tienen interacción interna entre ellos pero su visualización es individual.
En las siguientes imágenes, correspondientes a Ágora, podemos observar como tenemos el menú del conjunto de aplicaciones o ventanas que contiene junto con todas las opciones que nos brinda la aplicación. Es un menú desplegable donde cada rama es un conjunto de procesos agrupados por finalidad o tipo. Al seleccionar cualquier opción de este menú se nos abre una ventana (segunda imagen) con los parámetros que tiene asociados la persona que está utilizando la aplicación.
Tal como podemos observar las opciones y el conjunto de la información se muestra de forma conjunta dificultando la lectura de la información y de las propias opciones.

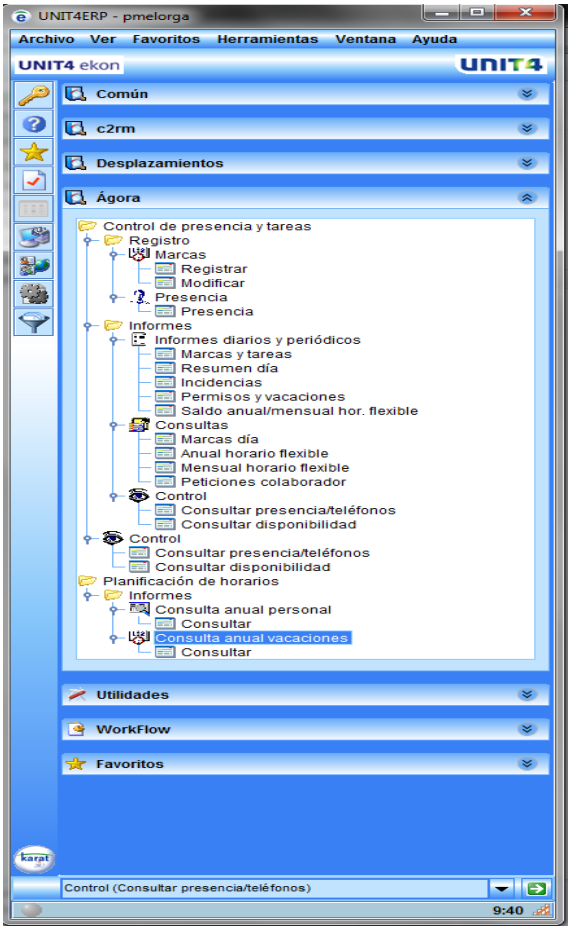


Figura 1: Imagen del contexto de la situación actual.

Al seleccionar una opción del menú contextual se superponen en orden de apertura, es decir, la última ventana abierta es la que primero se muestra. Esta condición hace difícil la tarea de un Gestor que debe tener varias ventanas abiertas a la vez para poder visualizar diferentes parámetros.
De esta forma el Gestor tiene la barra de tareas del sistema operativo un conjunto de ventanas que ha ido desplegando a medida que ha necesitado acceder a la información, cosa que dificulta encontrar la ventana que queremos visualizar o incluso dar pie a equivocación a la hora de seleccionar una misma venta.

No obstante la aplicación muestra la información correctamente y de una forma ordenada, lo que se busca es

mejorar la parte de interacción visualización para que el usuario pueda tener una visión global de la información y evitar que tenga que tenga que interactuar con la aplicación un tiempo excesivo ya que de un solo vistazo pueda visualizar los eventos más relevantes.

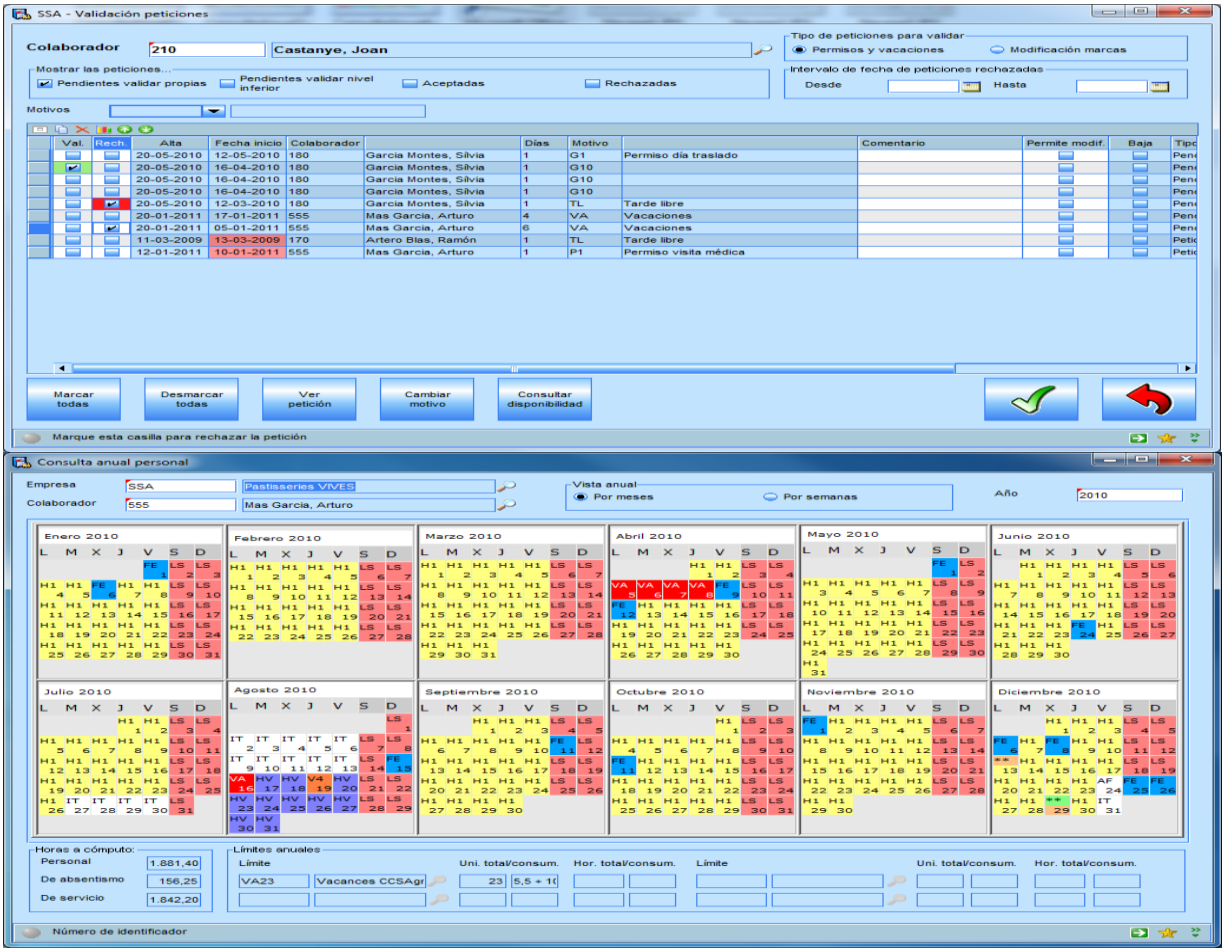


Figura 2: Imagen del contexto de la situación actual.

4.1.2. Diagnóstico del sistema actual

Aunque la aplicación es útil a la hora de gestionar un colaborador o un conjunto de colaboradores no satisface las necesidades de la compañía que se detallan a continuación.

Se han detectado las siguientes carencias:

- No permite la visualización al mismo nivel de varios formularios.
- Hay formularios que no permiten la visualización en conjunto de los diferentes colaboradores.
- No es posible mostrar la información más relevante o crítica en ese momento.

- Los formularios necesitan de una acción previa para mostrar los datos.
- La información se muestra en bloque, no hay una diferenciación entre parámetros relevantes en comparación a otros.

4.2. DEFINICIÓN DEL PROYECTO

4.2.1. "Escritorio".

Un Escritorio es una herramienta de karat que permite a un usuario añadir una serie de formularios insertados en ventanas o cajitas. Estas ventanas tienen la posibilidad de albergar cualquier tipo de formulario karat además de ofrecer una serie de características al usuario.
Permiten el libre movimiento y situación por el espacio del Escritorio siempre y cuando haya el espacio suficiente para ello.
Permite definir unos parámetros de entrada y de salida de las diferentes cajas. Un parámetro de entrada se refiere a que esa ventana depende de un ítem que proviene de otra ventana para poder mostrar información. El parámetro de salida define qué tipo de información de esa caja queremos que sea enviado a todas las cajas con las que tiene correspondencia. Cabe señalar que si eliminamos una caja con la que existe una relación de parámetros con otras, esos parámetros se los deberá insertar el propio usuario a mano.
El Escritorio dispone de un calendario global que puede ser el que gestione las diferentes ventanas que disponen de un campo fecha.
Otra característica fundamental es que el refresco de las ventanas se puede definir en segundos en la propia ventana por si alguna información relevante ha cambiado. También se puede de cambiar el color y el tamaño de la propia ventana.

Cabe señalar que todas las ventanas disponen de tres vistas: la minimizada, la estándar y la maximizada. La minimizada es un tipo de ventana que se muestra cuando tenemos otra ventana maximizada en total de la pantalla del Escritorio, por lo tanto esta se muestra en formato minimizado.
El formato estándar es la vista que se muestra al iniciar el Escritorio. Todas la ventanas al iniciarse tienen la vista estándar.
Y por último tenemos la vista maximizada, donde la ventana incluye el total de la pantalla del escritorio excepto una pequeña parte que permite que el usuario pueda visualizar las diferentes ventanas minimizadas.
El usuario debe diseñar las tres vistas, la estándar, maximizada y minimizada. Las tres tienen el mismo formulario de uso pero lo único que cambia es el nombre de las ventanas dentro del formulario y su denominación.

Para la estándar box_std, para la minimizada box_min y para la ventana maximizada ventana base.

A continuación se muestra la pantalla inicial de un escritorio, tal como se ha comentado antes, el usuario es el encargado de añadir las cajas que crea convenientes. En la figura 5, podemos observar el menú contextual que nos brinda la aplicación para poder insertar nuestras imágenes. Existen plantillas de escritorio que ya vienen con cajas predefinidas, el usuario puede decidir si eliminarlas, recolocarlas o cambiar sus parámetros. Tal como hemos comentado antes el escritorio es una herramienta totalmente personalizable.

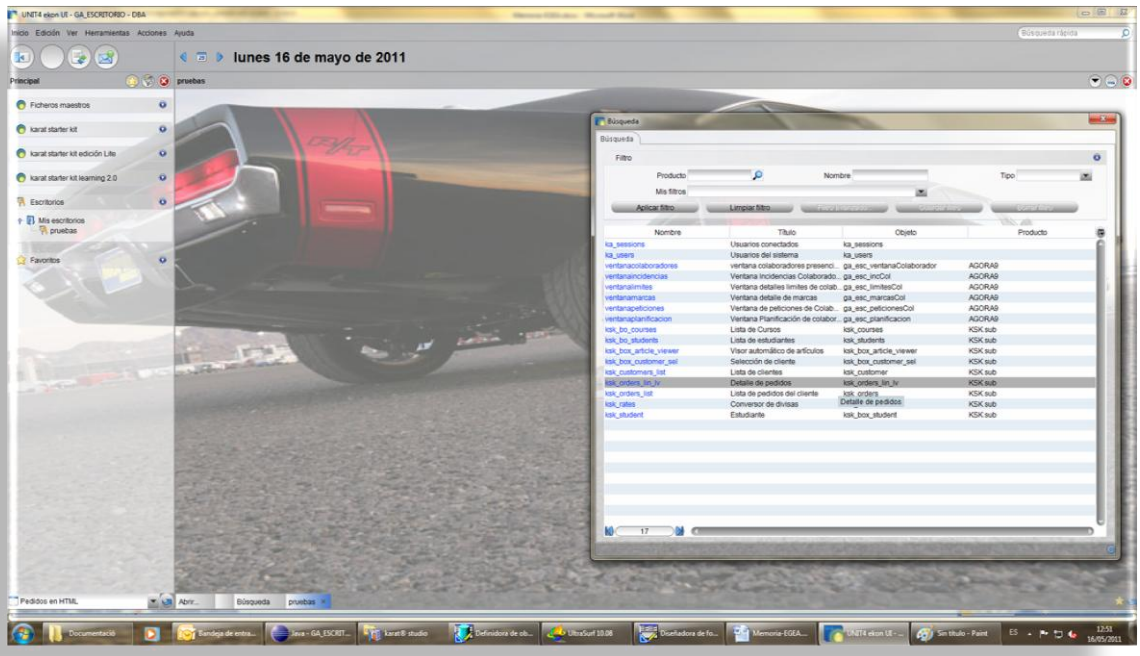


Figura 3: Imagen de un escritorio, pantalla inicial.

4.2.2. Descripción del producto a obtener

Deseamos obtener un escritorio interactivo para la visualización e interacción de la información. Debe de ser un escritorio capaz de mostrar a grandes rasgos la información más relevante así como los sucesos más importantes con repercusión inmediata en un entorno de trabajo o entorno laboral.
Mediante este sistema se permitirá que el gestor o responsable sea capaz de visualizar información en un período corto de tiempo y poder distinguir los eventos o problemas más importantes que pueden tener repercusión en un ambiente laboral.
Por otra parte el escritorio facilitaría la gestión de eventos, incidencias u otros elementos que el gestor deba tener en cuenta. El gestor puede modificar, validar o rechazar peticiones realizadas por el conjunto del personal que tiene a su cargo.

Toda la información está interrelacionada, es decir, en todo momento el gestor puede tener una vista en detalle de cada colaborador con sus determinadas características para un trato en detalle y particular, como una vista global que permita la visualización en conjunto de todos sus colaboradores. Este hito permite que el gestor pueda adoptar decisiones teniendo en cuenta el contexto global de los colaboradores que tiene a su cargo.

4.2.3. Descripción general del sistema

El sistema debe mostrar la información de todos los colaboradores en conjunto y también en particular permitiendo que el gestor en todo momento pueda visualizar de forma directa los puntos que requieren una mayor atención para su posterior solución. Los puntos o elementos críticos deben aparecer con mayor relevancia que los que no afectan a una posible mala coordinación de los diferentes colaboradores.

Por otra parte las acciones de modificar, rechazar o aceptar deben tener un sentido global y particular, es decir, se debe permitir una acción tanto en el conjunto de los colaboradores, como en el ámbito específico del colaborador.

Los cambios, ya sean modificaciones, aceptaciones o rechazos deben ser mostrados al instante dando al gestor una idea de conjunto sin implicar la interacción, si no es necesario, más allá del propio Escritorio.

La idea es que el gestor tenga toda la información en el conjunto del Escritorio y en caso particular deba interactuar en detalle con la ventana específica. Queremos obtener pocas interacciones máximo rendimiento.

4.2.4. Datos de entrada del sistema

Los datos de entrada al sistema son:

- Los diferentes colaboradores que dependen del gestor
- Calendarios laborables asociados a cada colaborador
- Tipos de jornada y horarios
- Marcas de cada colaborador
- Saldo de horas respecto a cada colaborador
- Peticiones de jornada prevista
- Incidencias de jornada procesadas

4.2.4. Funciones y/o resultado que ha de obtener el sistema

Funciones o resultados esperados:

- Agilización en la interacción con la aplicación
- Visualización directa de los parámetros más importantes. Leyendas de símbolos que definen el calendario laboral
- Consulta de los elementos más relevantes en el conjunto de los colaboradores
- Visión de conjunto y detalle de cada colaborador

- Interacción intuitiva
- Gestión de forma directa de los colaboradores
- Pasos simples para gestionar grandes cambios
- Todo cambio afecta al conjunto de la aplicación, también la selección.
- Diseño de tres tipo de vistas[3], la minimizada, maximizada y estándar.

4.2.5. Otras características del producto a desarrollar

Las características antes descritas son las más relevantes, es decir, la base del proyecto ha realizar. Por encima de esta base podemos determinar un conjunto de características no tan restrictivas.

- El usuario es capaz de modificar la disposición de los elementos que componen el escritorio
- Se pueden añadir más ventanas o elementos al escritorio general
- Colores intuitivos. Los casos que requieran mayor importancia deben ser en colores llamativos, el resto en colores suaves que no afecten a la visualización de la información
- Convenio de tamaños y estándares a la hora de representar ventanas, tamaño de letra e imágenes

4.2.5. Descripción del proyecto.

Definición de un escritorio gestor compuesto por varias ventanas[4] las cuales muestran un conjunto de información relevante y completa.
Basado en un diseño de plantilla de una cuadrícula de 12 casillas[5] x 12 casillas se integran los diferentes módulos o ventanas que lo componen. Cabe decir que el diseño debe ser ampliable para cuadrículas mayores o menores, a tales efectos la visualización de la información podría verse afectada.

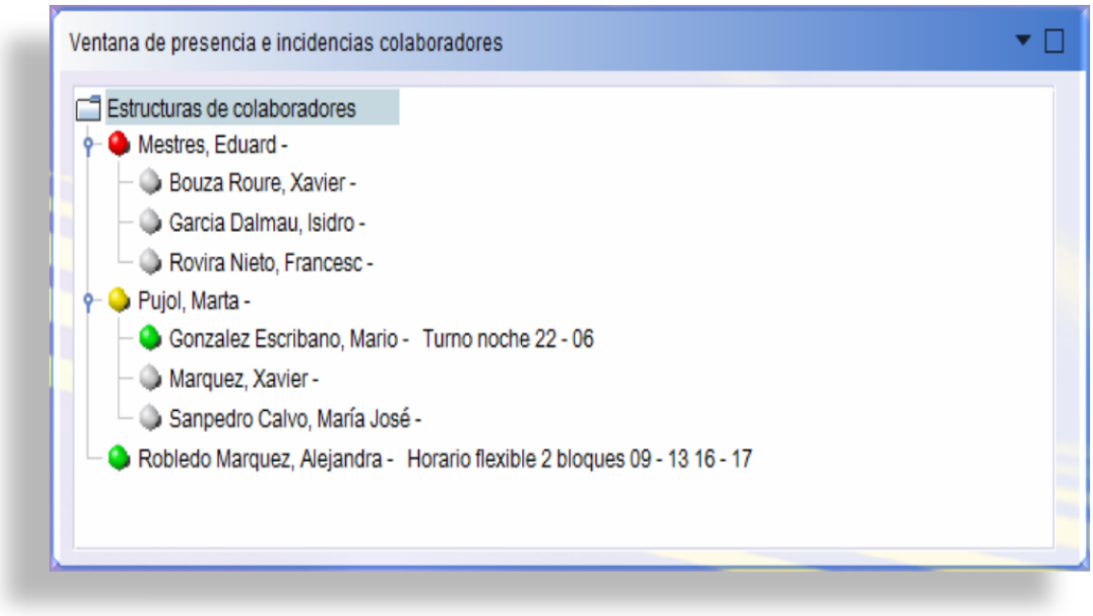
El escritorio está compuesto por un conjunto de 6 ventanas con una función y características diferentes entre ellas.
Se puede dar el caso que una misma pantalla contenga una o más ventanas dependientes de la ventana principal, en función de si queremos mostrar o no más información.

- Ventana de Control de Colaboradores. Esta ventana corresponde a la ventana principal donde se muestra el conjunto de colaboradores. En ella podemos visualizar en un árbol la posición que ocupa cada colaborador en la jerarquía de mando de una determinada empresa, puede darse el caso de que un

[3] Vista: corresponde a las diferentes visualizaciones que contiene una ventana del escritorio. Cada una determinada por un tamaño tal como indica su nombre.
[4] Ventana: en el ámbito del escritorio define un objeto en el cual se insertan formularios karat.
[5] Casilla: en el ámbito del escritorio define el número de parcelas o huecos que contendrá el escritorio para poder situar en el las diferentes ventanas creadas.

Gestor tenga otros gestores bajo su supervisión, con lo cual el árbol de jerarquía que se muestra es en función desde que colaborador queremos visualizar.
Cada colaborador dispone de un icono descriptivito, denominado presencia que indica mediante un color significante, si el colaborador, está presente, ausente o realizando una pausa, además de si el colaborador ya ha realizado su jornada habitual. Podría darse el caso que el colaborador realizase un horario nocturno y el gestor no.

A continuación del nombre del colaborador se muestran las diferentes notas o informaciones respecto a cada colaborador respecto al día actual. Dicha información corresponde al tipo de jornada del colaborador, la propia descripción o si tiene algún tipo de incidencia ya adjunta a su planificación que afecte a ese día, cómo podría ser una petición de día libre u horas libres. De esta forma se busca que esta ventana no sea de interacción sino al contrario que el Gestor pueda visualizar de forma directa qué colaboradores están presentes y el tipo de jornada e incidencias que tienen asociadas. Las vistas maximizada y estándar son las mismas a diferencia de la minimizada que muestra el total de colaboradores que están presentes sin realizar ninguna pausa en ese momento. La ventana se actualiza cuando el gestor pulsa la opción actualizar o en el momento que ingresa una marca en el sistema mediante la ventana de fichajes.



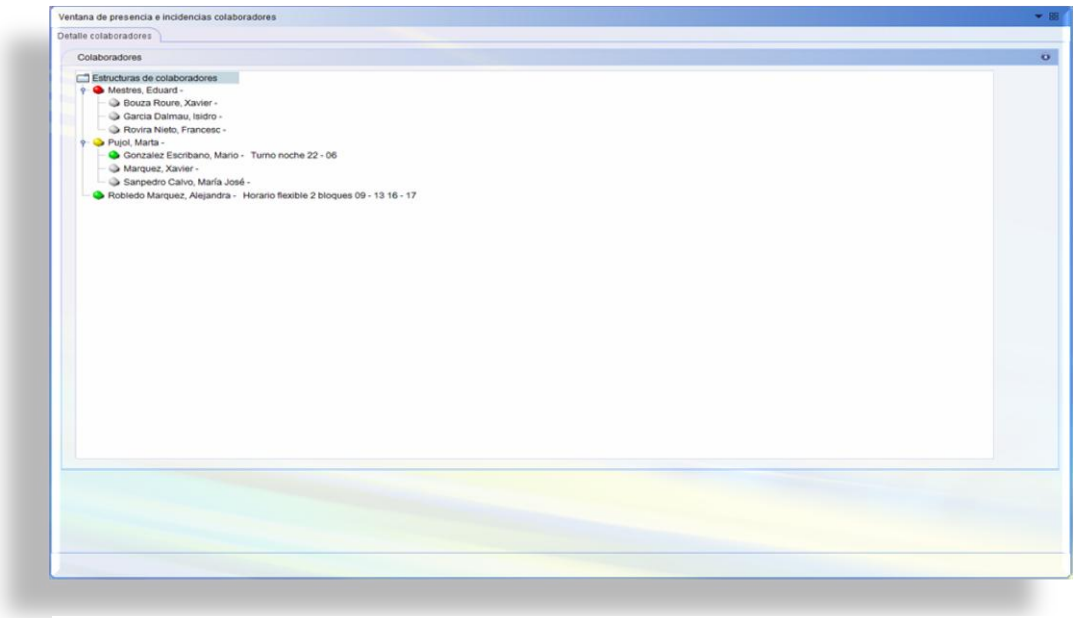


Figura 4: Imagen de la vista estándar y maximizada de la ventana colaboradores.

- Ventana de incidencias. En esta ventana se muestra el conjunto de incidencias que tiene el Gestor, como validador, por aceptar o rechazar. Ágora recopila el total de incidencias del total de días que tiene un Gestor por validar o rechazar. En caso de la ventana estándar se muestra los colaboradores, el tipo de incidencias que tienen y el número de incidencias asociadas a ese tipo. Esta vista sólo es de visualización.

  En la vista maximizada se muestran el total por colaborador, fecha, código, si está justificada, si tiene asociado un comentario, un parámetro, si fue anteriormente justificada y rechaza, etc. El colaborador dispone de un conjunto de filtros para poder visualizar la información, cómo el de tipo, origen, causa, motivo, etc. y de un filtro por colaborador separado del resto de filtros. En esta vista podemos visualizar el conjunto de incidencias enumeradas una por una por fecha y tipo que afectan a cada colaborador.
  Además se permite la opción de, al seleccionar una determinada incidencia, poder ver el detalle del día al que afecta, es decir, las horas en las cuales se ha producido. De esta forma el gestor puede visualizar el total de la información en una misma pantalla a más de poder aplicar los filtros que crea necesarios para su visualización.

  Otras opciones disponibles son poder marcar y desmarcar las diferentes incidencias para validarlas de forma conjunta o rechazarlas sin necesidad de ir una por una. Este hito permite validar o rechazar todas las incidencias pendientes de un determinado colaborador si se ha seleccionado previamente. Por último, en la ventana minimizada, se muestra el total de incidencias que tiene el Gestor por validar o rechazar.

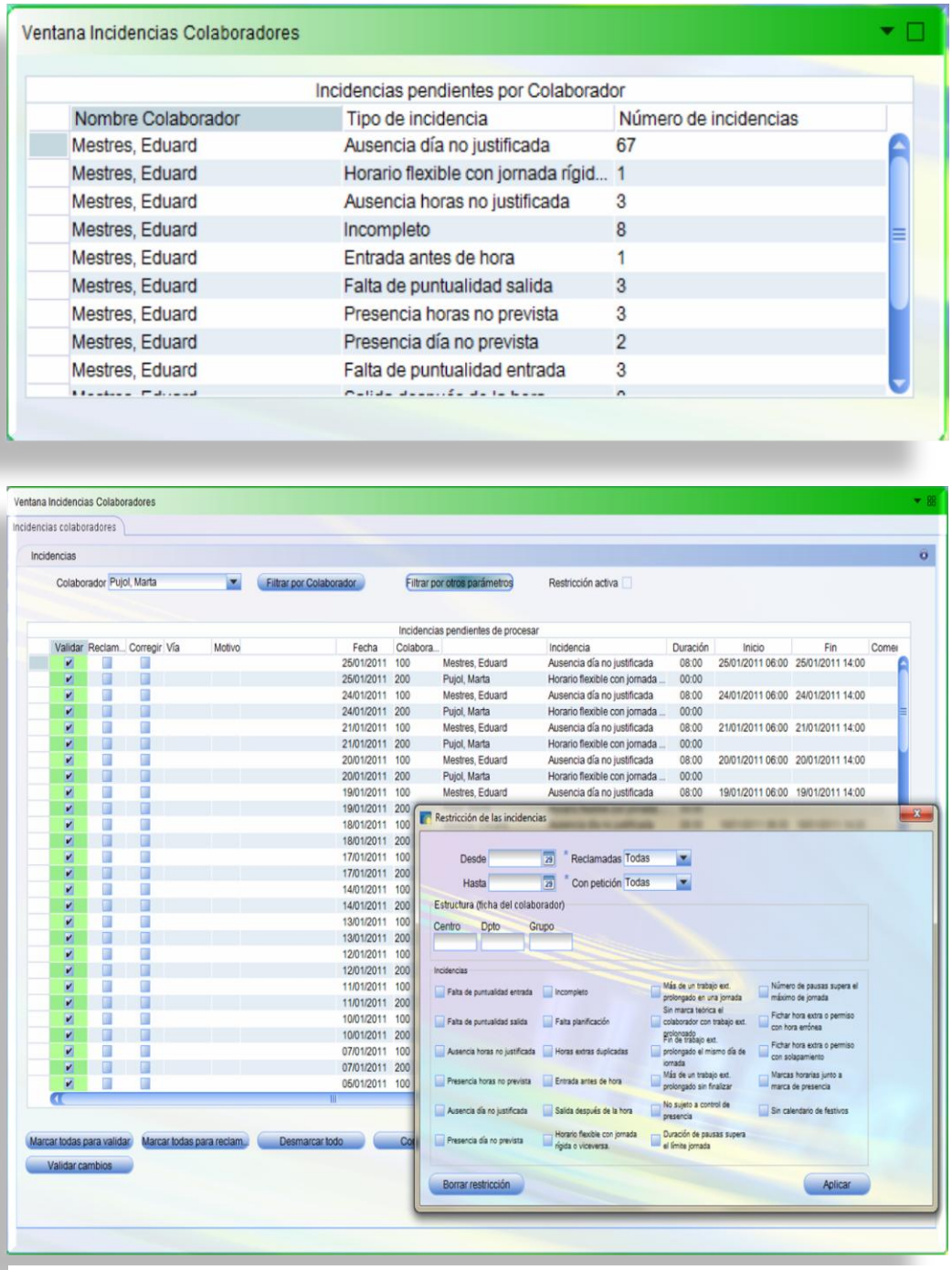


Figura 5: Imagen de la vista estándar y maximizada de la ventana de las incidencias de cada uno de los colaboradores.

- Ventana Peticiones. En esta ventana se muestra el conjunto de las peticiones que realizan los diferentes colaboradores que el gestor tiene a su cargo.
  En la vista estándar se muestra el colaborador, la fecha de ingreso de la petición, la fecha para la cual se solicita la petición y el tipo de petición.
  En la vista maximizada el gestor puede ver las diferentes peticiones que realizan los diferentes colaboradores que tiene a su disposición ordenadas por fecha. En este tipo de vista el gestor puede validarlas o rechazarlas. Disponemos de diferentes opciones para su visualización. Las opciones de las

que dispone el gestor son: mostrar las propias peticiones, cabe recordar que un Gestor puede generar peticiones propias para ser validadas por un superior inmediato, las propias de los colaboradores que tiene a su disposición, las ya aceptadas y las rechazadas. Además de estos filtros generales disponemos de la opción de mostrar las peticiones que atañen a permisos y vacaciones que afectan a una jornada de trabajo o a las que pertenecen a modificación de marcas. Cabe señalar que una marca hace referencia a un fichaje entrada, salida, pausa, etc. y que afecta a unas horas determinadas no a una jornada en concreto. Disponemos de la posibilidad de ingresar dos fechas para poder filtrar por días, semanas, meses o años las peticiones que se nos muestran en pantalla.

Las peticiones se muestran en una gran tabla o grid general donde disponemos de una serie de campos que se enumerarán a continuación donde podemos seleccionar o deseleccionar una serie de opciones. Cabe señalar que cada petición ocupa una línea dentro del grid.
Las opciones que se muestran son las siguientes: validar y rechazar una petición si marcamos uno de estos dos campos afectará a la decisión final. En el grid podemos visualizar la fecha de alta de la petición, la fecha de inicio de la misma, el colaborador que la ha generado, el número correlativo a los que afecta, la descripción de la petición, el comentario añadido por el colaborador a la hora de crearla, si es posible modificarla por parte del Gestor, si se puede dar de baja y si está pendiente o errónea.
Estas son las diferentes opciones y tipo de información que se muestra en el grid principal.

Por otra parte el Gestor dispone de la opción visualizar en detalle y visualizar en planing. Visualizar la petición en detalle nos muestra una nueva ventana que se nos abre prevaleciendo sobre las demás. Este ventana nos muestra el colaborador, la fecha para la que se solicita, el estado de la misma, el motivo (una descripción), las horas a las que atañe, las que ya tiene pendientes o adjudicadas (el día de inicio se toma como referente) y la secuencia de autorizadores. La secuencia de autorizadores son los colaboradores que según la jerarquía de mando de una determinada empresa tienen la potestad para poder validar o rechazar peticiones, la secuencia implica que cualquier colaborador que esté en orden superior al colaborador que realiza la petición puede validarla o rechazarla.

Por último disponemos de un calendario que nos muestra los días señalados que ese colaborador tiene realizadas peticiones validadas, el calendario permite ir visualizando meses y años e ir consultando las diferentes peticiones de ese mismo colaborador. Además disponemos de la vista en planing, que contiene las mismas opciones que en la vista en detalle pero a diferencia de la anterior el colaborador puede visualizar la planificación que tiene asociada desde el día que se solicita la petición hasta el fin de mes. Si el colaborador realiza cualquier cambio como rechazar, validar o modificar a una petición

que afecta a la planificación de un colaborador, la ventana de planificación se actualiza con los cambios realizados en la ventana de peticiones.

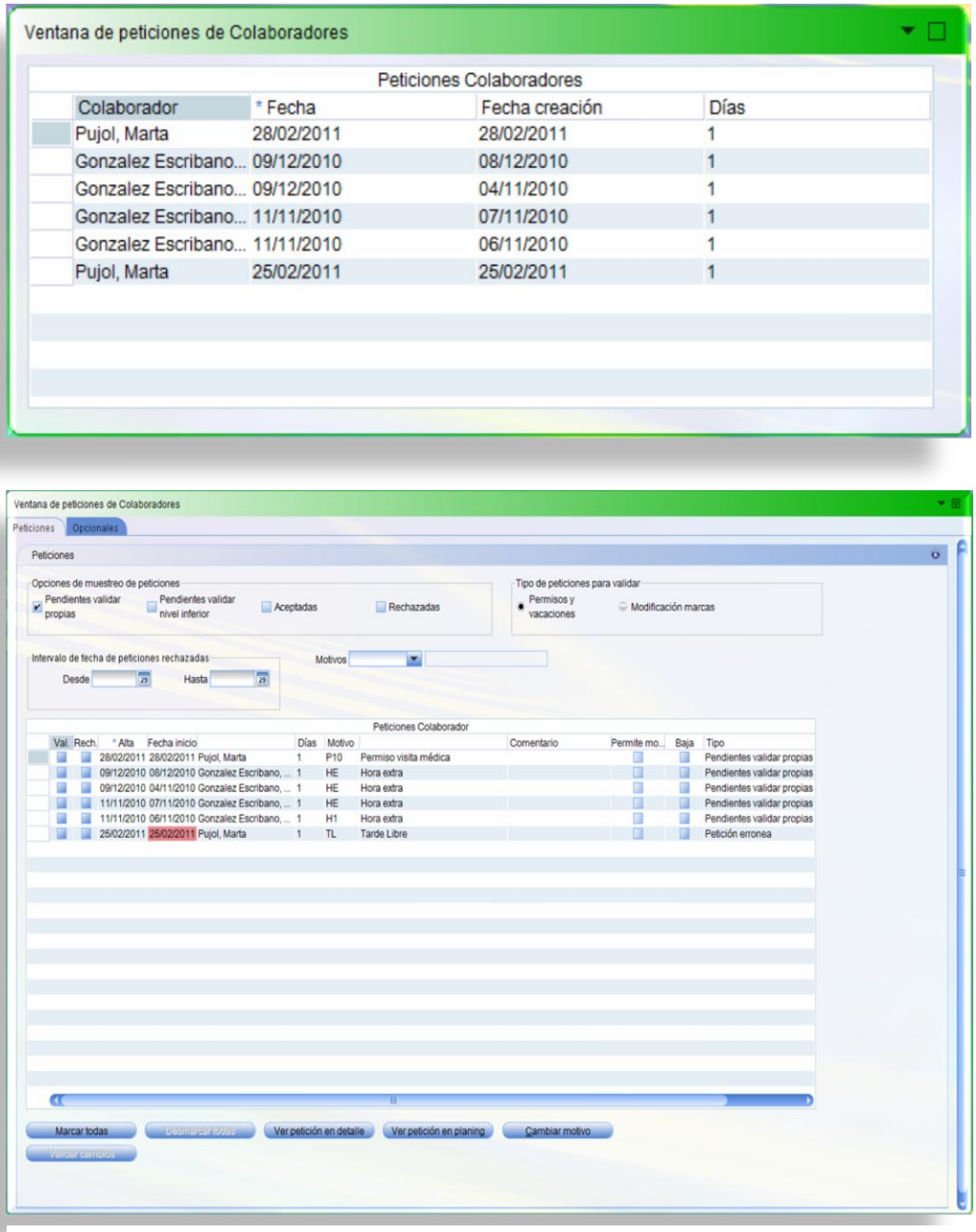


Figura 6: Imagen de la vista estándar y maximizada de la ventana peticiones por parte de los colaboradores y las diferentes opciones.

- Ventana de Planificación. Se encarga de mostrar la jornada que tiene un colaborador asociado a un día en un año concreto con las horas determinadas. En la ventana minimizada se muestra el año en el cual está centrada la planificación de los colaboradores ya que la planificación afecta a un año entero. En la ventana estándar tenemos los colaboradores, con su respectiva planificación asociada según el tipo de opción seleccionada para poder visualizar la información.

Esta ventana permite filtrar la planificación por horas, semanas y meses, además de mostrar la información en función del día seleccionado e un calendario. Para mejorar la visualización de la información permite que tanto el día y la hora se centren en la hora actual del día seleccionado. Otra característica importante es que la información se muestra mediante un código de colores de Ágora que pertenece a la jornada asociada que tiene cada uno de los colaboradores junto con un símbolo representativo de esa jornada asociada.

Además dispone de una opción que al pulsar sobre uno de estos recuadros se nos muestra información detallada, como es el tipo de jornada, el color que la representa, una breve descripción y las horas que lo componen, horas de entrada y salida, además de si el colaborador tiene algún tipo de incidencia incorporada a la planificación. El calendario permite ir navegando por los diferentes años, meses, semanas y días para visualizar la información. Por último la ventana maximizada permite obtener toda la planificación de los distintos colaboradores de un año entero.

Se ofrece la posibilidad de filtrar la información mediante dos fechas introducidas por parte del Gestor. Por último en la parte inferior de la propia ventana se describen el tipo de jornada que hay representada en la planificación, las horas que contiene cada una además de las incidencias al día que afectan con su respectivas horas y descripción. Por último la ventana se actualiza al realizar una modificación en la ventana de peticiones que afecten a la planificación de un colaborador.

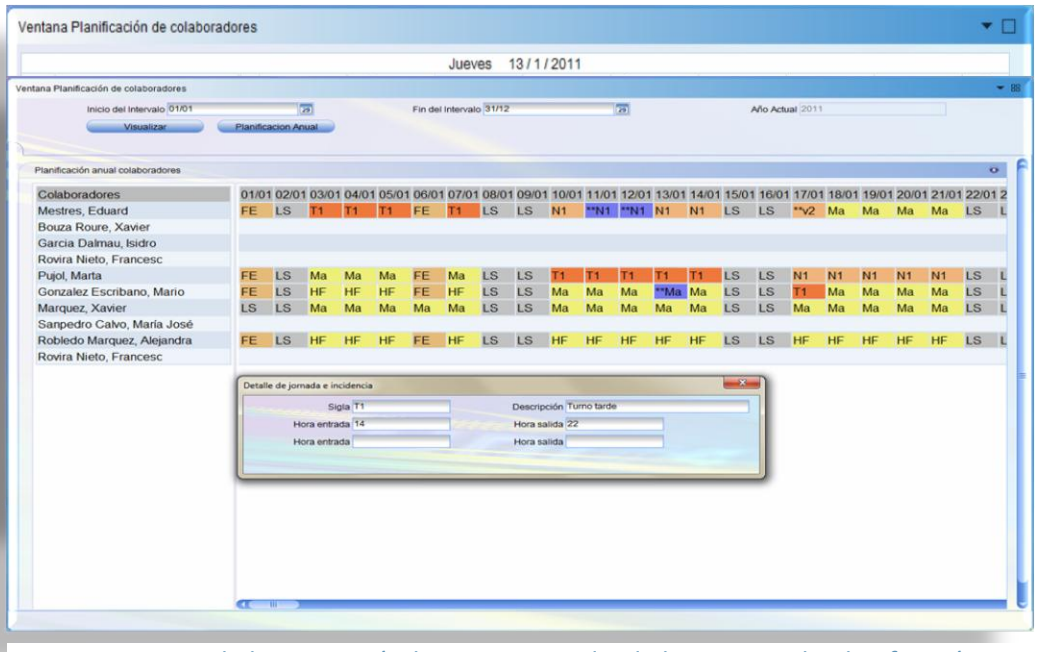


Figura 7: Imagen de la vista estándar y maximizada de la ventana de planificación asociada a cada uno de los colaboradores.

- Ventana de Límites. Muestra información de los límites que tiene asignado un determinado colaborador en un año así como las horas computadas de trabajo, de servicio y de disfrute. En la ventana minimizada disponemos del año actual en el cual estamos visualizando los diferentes límites de los colaboradores. En la ventana estándar se muestra el detalle de las horas de cómputo, de servicio y de disfrute que tiene un colaborador asociado.
  Al maximizar la ventana, se nos muestra una lista con todos los colaboradores y el año que queremos visualizar. Al seleccionar un colaborador de la tabla superior automáticamente se muestra la información referente a las unidades de los límites que tiene asignado el colaborador seleccionado. Las unidades del límite de agrupan en cuantas tiene te disfrute, cuantas ha consumido, se muestran, en formato horas y formato días.

Figura 8: Imagen de la vista estándar y maximizada de la ventana correspondiente a los límites que tiene asignado cada colaborador.

- Ventana de marcas. Permite realizar fichajes, peticiones, introducir marcas y justificantes de un determinado día al Gestor. La ventana estándar es la encargada de ofrecer dichas funcionalidades ya que muestra una lista con todos los colaboradores que el Gestor tiene a su cargo y tiene la posibilidad de realizar dichas opciones para el propio Gestor y para sus colaboradores.
  La ventana maximizada por el contrario permite consultar los fichajes de los respectivos colaboradores en un fecha seleccionada así como el tipo de horario que tiene asignado a esa fecha en concreto y las incidencias en el caso de tenerlas.

  Las funciones de las dos vistas son totalmente distintas, la estándar permite una interacción rápida ya que no es de consulta, al contrario, se brinda al Gestor la posibilidad de gestionar por su parte todos los fichajes, marcajes, peticiones e incidencias de todos los colaboradores que tiene a su cargo. De esta forma si un colaborador no puede realizar un marcaje por un motivo determinado el Gestor puede hacerlo en su lugar, lo mismo ocurre con las peticiones e incidencias. Por otra parte la vista maximizada es de consulta para saber el tipo de marcajes y la hora en la que se han realizado ya que de esta forma podemos comparar con la jornada e incidencias asignadas y poder determinar el grado de puntualidad de un determinado colaborador a más de sí está cumpliendo con el horario que tiene asignado y de esta forma si fuera necesario abrir una incidencias al colaborador que infringe dichos horarios.

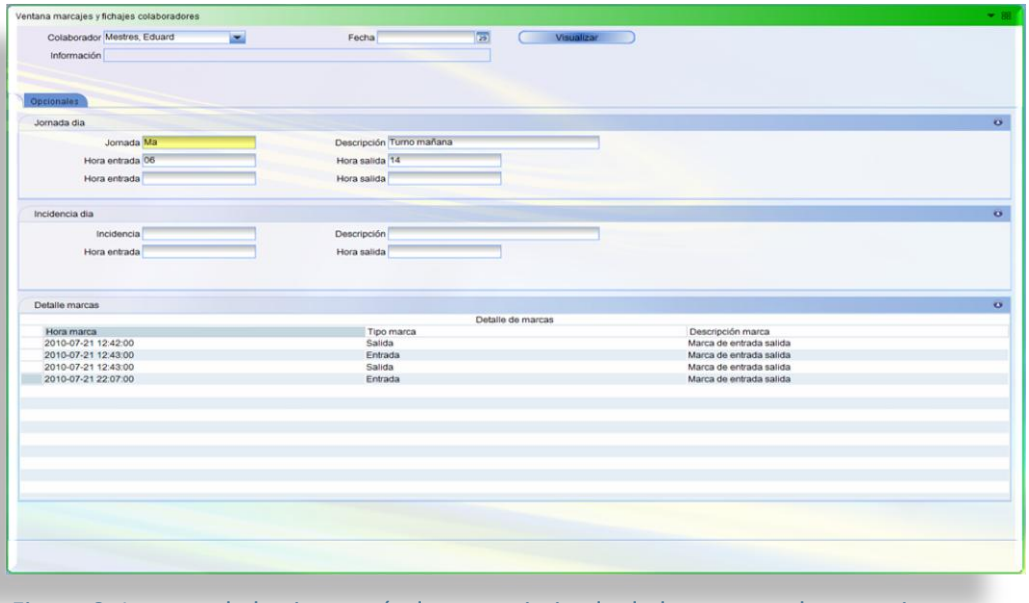


Figura 9: Imagen de la vista estándar y maximizada de la ventana de marcajes que tiene asignado cada colaborador.

4.2.6. Vista del Escritorio Gestor de Ágora.

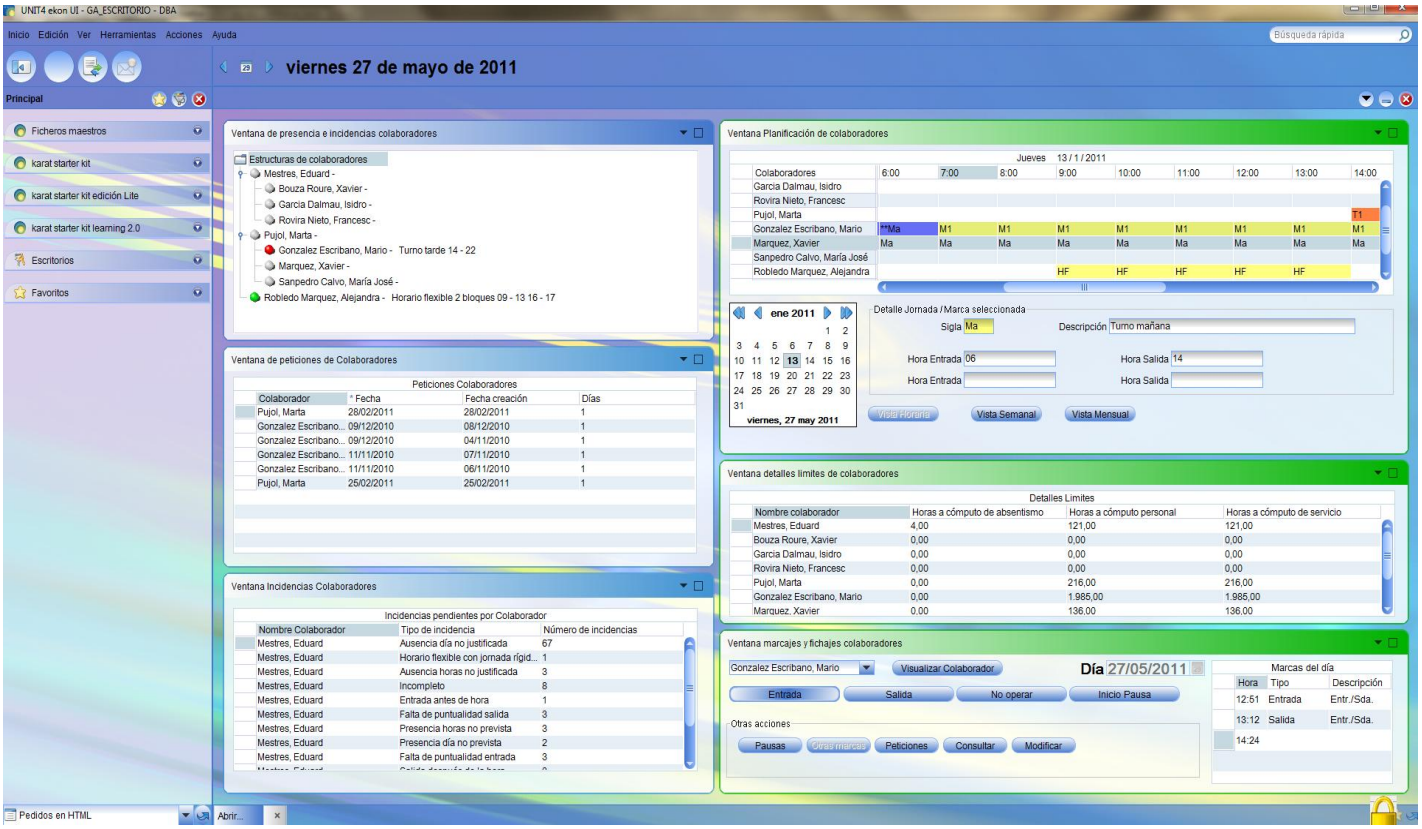


Figura 10: Imagen de la pantalla inicial del Escritorio Gestor de ekon Ágora.

4.3. REQUISITOS DEL PROYECTO

4.3.1. Requisitos funcionales

Lista de requisitos funcionales de la aplicación a desarrollar:

RF1. Creación de los formularios para cada una de las ventanas con sus respectivas vistas.
RF2. Visualización de presencia de colaboradores.
RF3. Visualización de planificación del conjunto de colaboradores.
RF4. Visualización de incidencias del conjunto de colaboradores.
RF5. Visualización de peticiones del conjunto de colaboradores.
RF6. Visualización de marcajes del conjunto de colaboradores.
RF7. Visualización de límites del conjunto de colaboradores.
RF8. Posibilidad de filtrar la información por colaborador.
RF9. Posibilidad de filtrar la información por fecha ya sea hora, día, mes o año.
RF10. Validar / Rechazar peticiones de los colaboradores.
RF11. Validar / Rechazar incidencias de los colaboradores.
RF12. Fichaje del propio Gestor o a un colaborador seleccionado.

4.3.2. Requisitos no funcionales

Exigencias de cualidades que se han impuesto al proyecto principalmente centradas en el diseño de la aplicación tanto a nivel visual como a nivel de interacción. En este caso se exige el uso de karat como plataforma tecnológica de desarrollo, que la aplicación siga un patrón visual de diseño de cada uno de los módulos o apartados que componen cada una de las ventanas, en nuestro caso Walnut, que el sistema permita un cierto margen de error por parte del usuario. La aplicación tiene que informar al usuario que se ha cometido un error a la hora de interactuar o introducir un determinado dato y por lo tanto se deber permitir la corrección del error o que el usuario pueda volver a introducirlo, sin que la aplicación finalice su ejecución.
A más la interfaz gráfica debe ser intuitiva y amigable, el usuario no tiene que ser un experto y no debe de tener dificultades a la hora de comprender los datos que muestra la aplicación, debe de ser una interfaz que con cada interacción permita al usuario su aprendizaje. Por último el software debe de ser ampliable en futuras versiones y la codificación del mismo tiene que permitir la evolución del software.

Debemos recordar que el software a desarrollar se tiene que basar en las necesidades del cliente, pero también ajustados a otros criterios, como el modelo de negocio de la compañía, los recursos disponibles y el tiempo de entrega. Es obvio que

software a desarrollar no solo debe cumplir con los requisitos funcionales (escribir código ajustado a los requisitos funcionales) sino también con las cualidades suplementarias (requisitos no funcionales) o de lo contrario no cumplirá con su misión: desarrollar el software que se necesita en el momento y condiciones que se tienen disponibles, o dicho de otra manera, desarrollar software de calidad.

Lista de requisitos no funcionales de la aplicación a desarrollar:

RNF1. El diseño ha de seguir el estilo Walnut.
RFN2. Utilizar las herramientas que proporciona Karat
RFN3. Tolerancia de errores y acciones incorrectas.
RFN4. Facilitar el trabajo al usuario mediante una interfaz amigable.
RFN5. El software ha de permitir ser ampliable.

4.3.3. Restricciones del sistema

El proyecto está enfocado a múltiples clientes por lo tanto a la hora de desarrollar la nueva plataforma se debe tener en cuenta una serie de limitaciones a la hora de crear la nueva aplicación.
Al tratarse de una aplicación desarrollada en la plataforma tecnológica karat debe acotarse a las funcionalidades que brinda dicha plataforma, todo aquello que karat no contemple no podrá ser implementado.
Al tratarse de un software para la empresa UNIT4 debe seguir los patrones y estilos de codificación definidos por la empresa así como las reglas de visualización de la propia empresa. La empresa es quien impone dichos patrones puesto que sus productos aunque sean de diferente funcionalidad y objetivo deben de seguir los mismos patrones para dar sensación de homogeneidad.
Por último el proyecto debe estar finalizado antes del 30 de Junio de 2011 requisito impuesto por parte de la Universidad y de la empresa UNIT4 para el desarrollo de la aplicación.

Lista de las estricciones del sistema:

1. Utilización del software y del sistema de karat
2. Seguir los estándares de UNIT4.
3. El proyecto ha de estar finalizado antes del 30 de Junio de 2011.

4.3.4. Catalogación y priorización de los requisitos

#Req	Esencial	Condicional	Opcional
RF1	×		
RF2	×		
RF3	×		
RF4	×		
RF5	×		
RF6	×		
RF7	×		
RF8		×	
RF9		×	
RF10			×
RF11			×
RF12			×
RNF1		×	
RNF2		×	
RNF3		×	
RNF4			×
RNF5		×	

Tabla 4: Tabla de catalogación y priorización de los requisitos.

4.4. ALTERNATIVAS Y SELECCIÓN DE LA SOLUCIÓN

4.4.1. Solución propuesta

Utilizar un software que permita la creación de escritorios interactivos y que permita las funcionalidades descritas en el apartado 5.2.1 referente al Escritorio. Actualmente en el mercado no se dispone de ninguna herramienta que permita la creación de escritorios multifunción con las características antes descritas. En nuestro caso no se contemplan varias alternativas que permitan una comparación entre las mismas para una posterior comparación.
Tal como se ha comentado antes se debe desarrollar un software bajo las directrices de UNIT4 y basado en la plataforma tecnológica karat, por lo tanto no es posible añadir ,en caso que lo hubiera, una aplicación de otras compañías ya que iría en contra de las restricciones del sistema.

Por lo tanto de propone desarrollar un nuevo módulo para el producto ekon Ágora desde 0, adaptándolo a todas las necesidades y cubriendo todos los requisitos que se nos indican.

En este caso el coste sería 0€, implica un mayor trabajo por parte del desarrollador pero asegura que todos los requisitos serán satisfechos.
Implica la creación de nuevos formularios y en todo caso adaptar el modelo de datos existente a la nueva aplicación a crear.

4.4.2. Solución propuesta, características

Al no disponer de alternativas para el desarrollo de la nueva aplicación, puesto que el mercado no ofrece una aplicación con las funcionalidades esperadas del escritorio y que no esté vinculada a otra compañía de la competencia, la solución propuesta que implica el desarrollo de la nueva aplicación desde 0 implica un conjunto de costes y requerimientos.

Los costes de adquisición son nulos, puesto que se desarrolla desde 0, no es necesario adquirir software de otras compañías o archivos fuente para el desarrollo. El coste de adaptación es bajo puesto que la aplicación se desarrolla bajo la plataforma tecnológica karat y representa que el desarrollo de la nueva aplicación tiene que adaptarse a las funcionalidades y limitaciones que ofrece Karat. El coste de exploración para el desarrollo de la nueva aplicación implica que debemos aprender las funcionalidades que nos brinda karat y por lo tanto el coste es bajo. Tanto el coste de desarrollo y los ajustes de requerimientos son altos, por una parte se está creando una aplicación desde 0 y esto implica un coste añadido que si ya se tuviera una software base para ir desarrollando sobre esa base el producto final y por otra parte los requerimientos los marca el cliente y la propia empresa con los cual el ajustes de los mismos es alto.

En la siguiente comparativa se ponen en común los diferentes aspectos más relevantes

	Coste adquisición	**Coste adaptación**	**Coste exploración**	**Coste desarrollo**	**Ajuste requerimientos**
Solución	0€	bajo	bajo	alto	alto

Tabla 5: Tabla de costes de la solución.

4.5. PLANIFICACIÓN DEL PROYECTO

4.5.1. Recursos del proyecto

Aquí se presentará todo lo referente a la planificación del proyecto, también de que recursos se dispone para realizarlo.

El coste de los recursos humanos es aproximado, ya que por este proyecto el alumno recibirá un total de 2.352€. El coste ficticio del proyecto supondría alrededor de unos 25.000€.

- Recursos humanos:

Función	Coste/h
Jefe de proyecto	100€/h
Analista	50€/h
Programador	30€/h
Técnico de pruebas	20€/h

Tabla 6: Tabla de recursos humanos del proyecto.

- Recursos materiales:
  - Sistema de datos de Ágora
  - Licencia Karat
  - Maquinaria:
    PC – Intel Core 2Q8400 @ 2.66GHz
    4,00 GB RAM
    Disco duro 500GB
    Conexión a internet
    Pc Servidor Ottols Intel Core 2Q8400 @ 2.66GHz
    4,00 GB RAM
    Disco duro 500GB
    Conexión a internet
  - Software:
    Windows 7
    Eclipse
    Karat Studio[6]
    Karat Escritorio[7]

[6] Karat Studio: permite generar los objetos de karat, tales como objetos de negocio, formularios, etc..
[7] Karat Escritorio: aplicación que permite generar los escritorios y acceso a la funcionalidad de cada uno de los formularios u aplicaciones.

4.5.2. Planificación, fases del Proyecto

El proyecto se desarrollará de Noviembre de 2010 hasta Junio de 2011, con una dedicación de aproximadamente 20h semanales y un total de 560h.

Fecha de inicio: 8 de Noviembre del 2010.
Fecha de finalización prevista: 2 de Junio del 2011.

Este calendario puede variar ya que está planificado para hacer una jornada de 4 horas diarias.
Todas las fases del proyecto se desarrollan mediante un modelo lineal, por lo tanto , cada fase no comienza hasta que no se ha terminado la fase anterior. Se ha divido el proyecto en 12 fases. Las dos primeras corresponden al aprendizaje de la plataforma tecnológica karat y al estudio del contexto y situación actual. El resto de fases comprendidas desde la tercera hasta la decimocuarta corresponden al diseño de casa una de las ventanas de la aplicación así como las pruebas que se realizan individualmente de cada ventana.
Por último las dos últimas fases corresponden al seguimiento final de la aplicación y su comportamiento en conjunto con todas las ventanas ya operativas y a la documentación del proyecto.

Fases del proyecto son:

- Fase 1: Análisis del sistema actual de Ágora y del nuevo Escritorio gestor.
- Fase 2: Estudio del sistema de datos actual y si es necesario añadir tablas de datos para la implementación del escritorio.
- Fase 3: Implementación de la ventana de presencia de colaboradores.
- Fase 4: Pruebas y resultados de la ventana presencia de colaboradores
- Fase 5: Implementación de la ventana de planificación de colaboradores.
- Fase 6: Pruebas y resultados de la ventana presencia de colaboradores.
- Fase 7: Implementación de la ventana de incidencias.
- Fase 8: Pruebas y resultados de la ventana incidencias.
- Fase 9: Implementación de la ventana de peticiones.
- Fase 10: Pruebas y resultados de la ventana peticiones.
- Fase 11: Implementación de la ventana de límites.
- Fase 12: Pruebas y resultados de la ventana límites.
- Fase 13: Implementación de la ventana de marcajes.
- Fase 14: Pruebas y resultados de la ventana marcajes.
- Fase 15: Pruebas del conjunto de la aplicación.
- Fase 16: Documentación del Proyecto.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesora	Recursos
1	**Escritorio Gestor para el producto Ágora**	**149,5 días**	**lun 08/11/10**	**vie 03/06/11**		
2	**Curso de Formación**	10 días	lun 08/11/10	vie 19/11/10		Programador
3	**Planificación**	**6 días**	**lun 08/11/10**	**lun 15/11/10**		
4	Estudio de viabilidad	2 días	lun 08/11/10	mar 09/11/10		Jefe de proyecto; Analista
5	Aprobación Estudio de Viabilidad	1 día	mié 10/11/10	mié 10/11/10	4	Jefe de proyecto
6	Plan del Proyecto	3 días	mié 10/11/10	vie 12/11/10	4	Jefe de proyecto; Analista
7	Aprobación Plan de Proyecto	1 día	lun 15/11/10	lun 15/11/10	6	Jefe de proyecto
8	**Análisis de la aplicación**	**14 días**	**mar 16/11/10**	**vie 03/12/10**		
9	Reunión con el departamento de Ágora	1 día	mar 16/11/10	mar 16/11/10	3	Analista
10	Análisis de requisitos	2 días	mié 17/11/10	jue 18/11/10	9	Jefe de proyecto; Analista
11	Análisis del producto Ágora. Módulos de planificación horaria y control de presencia	7 días	vie 19/11/10	lun 29/11/10	10	
12	Documentación del Análisis	3 días	mar 30/11/10	jue 02/12/10	11	Analista
13	Aprobación del Análisis	1 día	vie 03/12/10	vie 03/12/10	12	Jefe de proyecto
14	**Diseño de la Aplicación**	**28 días**	**lun 06/12/10**	**mié 12/01/11**		
15	Estudio de las funcionalidades del programa generador de escritorios	10 días	lun 06/12/10	vie 17/12/10	8	Analista[50%]; Programador[50%]
16	Diseño de las ventanas del Escritorio Gestor	10 días	lun 20/12/10	vie 31/12/10	15	Analista[50%]; Programador[50%]
17	Diseño de los test de pruebas	5 días	lun 03/01/11	vie 07/01/11	16	Jefe de proyecto[50%]; Programador
18	Documentación del diseño	2 días	lun 10/01/11	mar 11/01/11	17	Analista
19	Aprobación de diseño	1 día	mié 12/01/11	mié 12/01/11	18	Jefe de proyecto
20	**Desarrollo de la Aplicación**	**86,5 días**	**jue 13/01/11**	**vie 13/05/11**		
21	Preparación del entorno de desarrollo	1 día	jue 13/01/11	jue 13/01/11	14	Programador
22	Creación de la ventana presencia de colaboradores	5 días	vie 14/01/11	jue 20/01/11	21	Programador
23	Test de la ventana presencia de colaboradores	1,5 días	vie 21/01/11	lun 24/01/11	22	Técnico de pruebas
24	Creación de la ventana incidencias de colaboradores	6 días	lun 24/01/11	mar 01/02/11	23	Programador
25	Test de la ventana presencia de colaboradores	3 días	mar 01/02/11	vie 04/02/11	24	Técnico de pruebas

26	Creación de la ventana peticiones de colaboradores	6 días	vie 04/02/11	lun 14/02/11	25	Programador
27	Test de la ventana peticiones de colaboradores	3 días	lun 14/02/11	jue 17/02/11	26	Técnico de pruebas
28	Creación de la ventana planificación de colaboradores	20 días	jue 17/02/11	jue 17/03/11	27	Programador
29	Test de la ventana planificación de colaboradores	10 días	jue 17/03/11	jue 31/03/11	28	Técnico de pruebas
30	Creación de la ventana limites de colaboradores	6 días	jue 31/03/11	vie 08/04/11	29	Programador
31	Test de la ventana límites de colaboradores	3 días	vie 08/04/11	mié 13/04/11	30	Técnico de pruebas
32	Creación de la ventana marcas de colaboradores	15 días	mié 13/04/11	mié 04/05/11	31	Programador
33	Test de la ventana marcas de colaboradores	7 días	mié 04/05/11	vie 13/05/11	32	Técnico de pruebas
34	**Test y pruebas**	**8 días**	**vie 13/05/11**	**mié 25/05/11**		
35	Pruebas de integración	2 días	vie 13/05/11	mar 17/05/11	20	Técnico de pruebas
36	Pruebas de estrés	2 días	mar 17/05/11	jue 19/05/11	35	Técnico de pruebas
37	Documentación de las pruebas	2 días	jue 19/05/11	lun 23/05/11	36	Técnico de pruebas
38	Aprobación de las pruebas y resultados	1 día	lun 23/05/11	mar 24/05/11	37	Jefe de proyecto
39	Corrección de errores	1 día	mar 24/05/11	mié 25/05/11	38	Programador
40	**Generación de la Documentación**	**5 días**	**mié 25/05/11**	**mié 01/06/11**	**34**	**Analista**
41	**Cierre del proyecto**	**1 día**	**mié 01/06/11**	**jue 02/06/11**	**40**	**Analista; Jefe de proyecto**
42	**Defensa del Proyecto**	**1 día**	**jue 02/06/11**	**vie 03/06/11**	**41**	**Analista**

Tabla 7: Tabla de tareas.

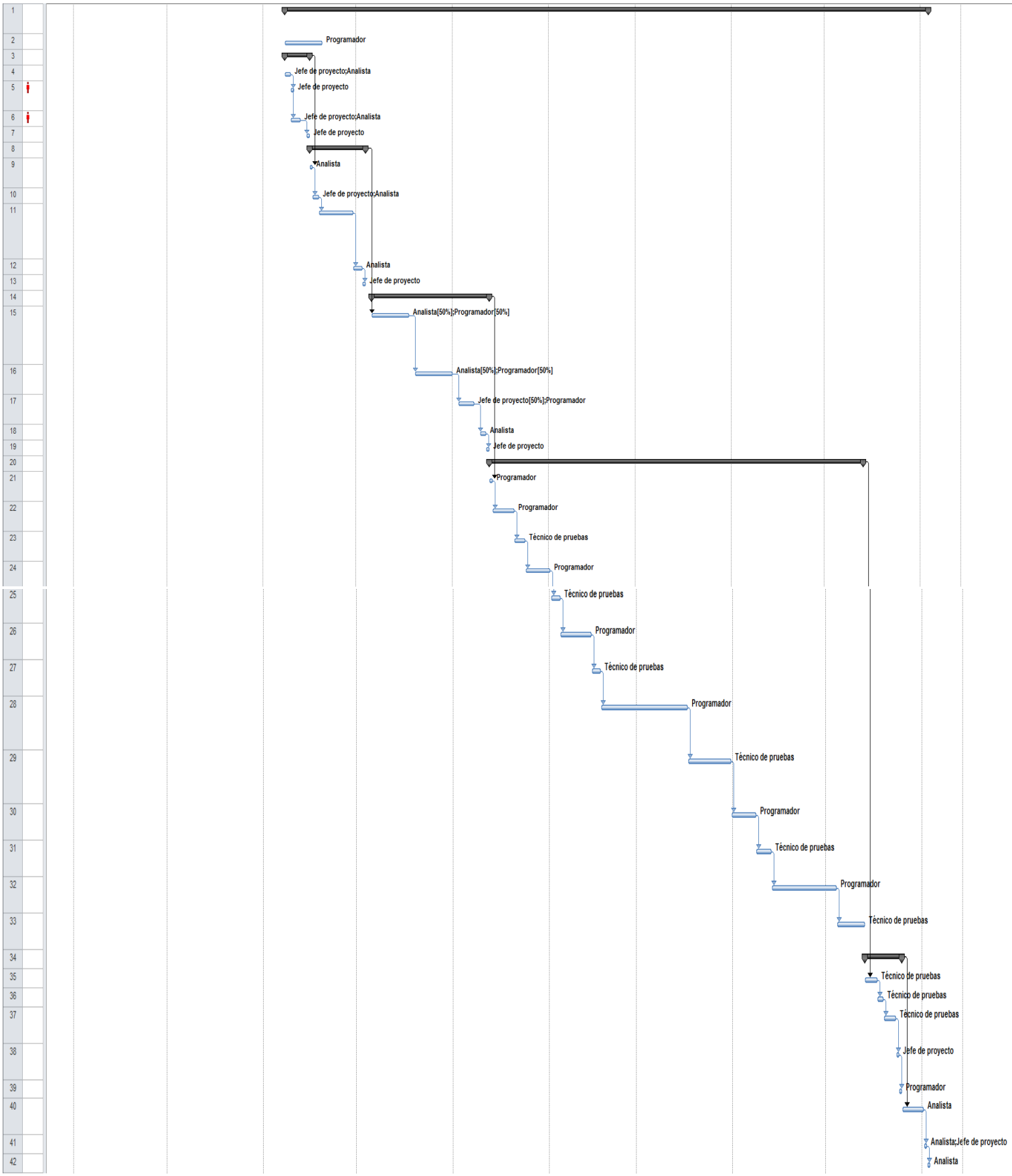


Figura 11: Diagrama de Gantt de la planificación anual.

4.6. PRESUPUESTO

4.6.1. Estimación coste de personal

A continuación se enumeran los costes de personal previstos, estos costes son aproximados.

Función	Coste/h	Horas	Coste total
Jefe de proyecto	100€/h	80	8000 €
Analista	50€/h	120	6000 €
Programador	30€/h	320	9600 €
Técnico de pruebas	20€/h	40	800 €
TOTAL		560	24400 €

Tabla 8: Tabla con los costes estimados de personal.

4.6.2. Estimación coste de los recursos

La siguiente tabla muestra los costes de los recursos necesarios para desarrollar el proyecto.

Recurso	Coste
PC	600€

Tabla 9: Tabla con los costes estimados de los recursos.

4.6.3. Resumen y análisis coste beneficio

Detalle de coste:

Coste personal……	24.400
Coste recursos……	600
TOTAL……	--------
…….	-
	25.000 €

Tabla 10: Tabla resumen y análisis coste beneficio.

El proyecto tiene un coste razonable si miramos las licencias de los productos actuales que se asemejan en el mercado Igualmente como hemos comentado antes esto es un coste ficticio, el coste real son los 2.352€ que percibirá el alumno.
Por este motivo aún se pueden apreciar en mayor medida los beneficios que se producirán al insertar el software dentro de la herramienta karat.

4.7. CONCLUSIONES

4.7.1. Beneficios e inconvenientes

Los beneficios que se buscan con el desarrollo de la nueva aplicación se pueden diferenciar en cuatro grandes bloques.
Por una parte el control eficiente del conjunto de colaboradores, tal como se ha comentado en la introducción del proyecto se busca que Gestor pueda tener en todo momento la visión de todos los cambios que realiza sin tener que abrir aplicaciones auxiliares. Visualización de la información de forma conjunta, la información esta agrupada y no dispersa. Esta es la principal característica del Escritorio Gestor, tener en una misma ventana aquellos procesos que son usados con más frecuencia por parte del Gestor agrupados en función de su objetivo. Crear una nueva aplicación para mejor e innovar el producto Ágora, puesto que con esta herramienta se intenta subsanar las deficiencias que tiene el sistema actual.
Por último los eventos de los factores más relevantes y aquellos que implican una atención inmediata ya que afectan a la toma de decisiones tengan una mayor visualización ya que permite una mayor gestión y control de los eventos más relevantes.

Por lo que se refiere a inconvenientes solo ha sido detectado uno, que en nuestro caso ha sido la necesidad de un periodo de formación para entender el uso de la plataforma tecnológica karat.
Si tenemos en cuenta los beneficios y los inconvenientes el **Proyecto es viable.**

FASE DE DISEÑO

5.1 INTRODUCCIÓN

5.1.1. Relación entre karat y Ágora.

Tal como hemos comentado antes, los productos de Unit4, en concreto Ágora, se desarrollan en la plataforma tecnológica karat. Esto implica que este producto está construido con las herramientas que nos proporciona la propia plataforma.
Karat está basado en un modelo de datos, en una forma de acceder a esos datos y como se representan. A continuación se describen estos procesos.

Los datos se almacenan en tablas dentro de un repositorio. Las tablas forman la base fundamental de karat, donde se almacenan los datos físicamente para poder ser tratados en un futuro. El repositorio es un objeto que contiene a parte de tablas, el conjunto de datos que permite que la aplicación funcione, como formularios, objetos de negocio, consultas, etc.
Más adelante entraremos en detalle en cada uno de los componentes. Cabe señalar que cada producto tiene un repositorio asignado, donde el modelo de datos es el mismo, lo único que varía son los datos que contiene cada una de las tablas, pues la configuración es la misma.

5.1.2. Descripción de los Objetos de Karat, que participan en la creación del Escritorio.

- Tablas: contienen los datos físicos que se mostrarán en la aplicación. Se definen los campos, el tipo de datos, las claves primarias, secundarias que contienen las propias tablas, el dominio de cada uno de los campos y la relación entre cada una de ellas.

- Consulta base: podemos definir un proceso que afecta a una o varias tablas por las cuales podemos acceder al conjunto de datos formado por cada uno de los campos qué contiene las tablas permitiendo qué los datos de una tabla filtren los mismos en otra, a más de poder seleccionar que datos de que campos concretos queremos seleccionar.

- Objeto de negocio: tal como su nombre indica es un objeto en el cual se definen una serie de campos con su tipo de datos. Estos campos pueden estar sin vincular, es decir, no mostrarán nada a no ser que se les asigne mediante código o pueden estar vinculados a los campos de una consulta base. Además, estos

campos, si están vinculados, permiten que el resultado sea filtrado por otra consulta base o que el resultado sea en función de los datos de una lista de sistema o de una variable global (se definirán a continuación). El objeto de negocio está formado por paneles, en los cuales podemos ir organizando los datos. Un panel puede contener los datos de una consulta base, otro panel los de otra y así respectivamente, permitiendo la separación del conjunto de campos.

- Lista de sistema: es una lista de Karat en la cual se define un índice y cada valor de ese índice será el que se obtendrá al acceder a dicho índice.

- Variable global: es una variable en la cual se define el tipo de dato que debe almacenar. Su valor puede ser común a varias tablas, consulta base, lista de sistema, objeto de negocio que necesiten ese valor para filtrar los datos, como complementario o por el hecho de mostrar siempre ese valor. Permite que no se borre nunca si seleccionamos la opción de guardar en repositorio o que sólo se mantenga el valor mientras es utilizada en una sesión de conexión con el servidor.

- Mensaje de Karat: permite definir un módulo de mensajes que contiene una serie de mensajes que serán mostrados en un momento determinado. Se puede definir, el cuerpo del mensaje (texto a mostrar), el icono a representar, las acciones a realizar y los botones que queremos que aparezcan.

- Formulario: es un objeto que relaciona a los objetos de negocio con sus respectivos paneles. Es la parte gráfica de la aplicación donde se colocan los campos del objeto de negocio de forma agradable visualmente siguiendo una serie de directrices, en ese caso walnut. En esta parte los campos del objeto de negocio pasan a llamarse controles puesto que cada campo puede ser de un tipo de control diferente. Los controles más habituales y empleados en el diseño de Escritorio han sido: los botones, las listas desplegables, los campos de texto, las semánticas [8], las cajas html[9], los grid o tablas de datos (un campo del objeto de negocio pertenece a una columna del grid) y el calendario.

Estos son los objetos más representativos que han sido utilizados durante la creación del Escritorio. Cabe señalar que en algunos casos ha habido que utilizar algunos de éstos ya existentes en el repositorio o crear de nuevos objetos, en el apartado de codificación se definirán los objetos que se han utilizado para llevar a cabo la realización del Escritorio.

[8] Semántica: se refiere a un campo de texto en el cual nos aparece un icono para que podamos seleccionar un valor de una lista o un calendario permitiendo al usuario mayor facilidad en su uso.
[9] Caja html: permite al usuario mediante código Java la inserción de una página web en HTML 4.0

Una característica común de estos objetos es que son multilenguaje. Esto implica que al desarrollador y programador un ahorro de faena a la hora de diseñar una nueva aplicación, ya que de esta forma el departamento de documentación y traducción puede realizar su trabajo sin realizar modificaciones en la estructura de los datos.

Todos los objetos antes descritos pueden ser modificados mediante una clase de personalización en código Java. Esta clase de personalización debe utilizar las librerías de Karat que son las que contienen las funciones específicas para poder modificarlos.

5.1.3. Relación entre los diferentes objetos de Karat.

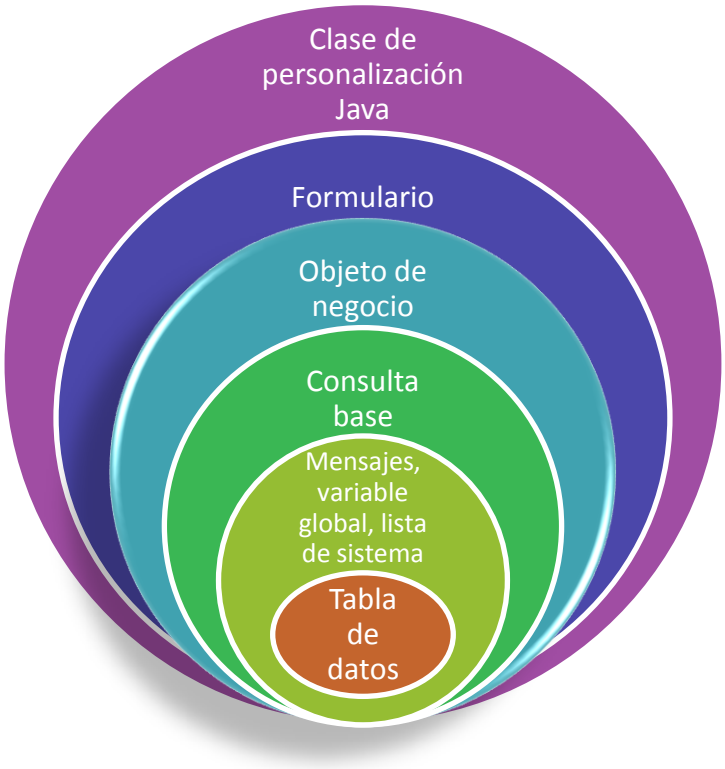


Figura 12: Relación entre los diferentes objetos de Karat.

5.1.4. Creación de objetos de Karat.

La creación de objetos de Karat se puede hacer de forma manual, es decir, el usuario define el conjunto de registros, la relación entre ellos, el comportamiento, el conjunto de datos, dominio, etc. o mediante asistente. El asistente permite crear estos objetos mediante una ayuda interactiva, pensada en los asistentes de Windows.
Para desarrollar los objetos se ha decidido que hacerlo de forma manual ya que permite que el desarrollador tenga una mayor libertad a la hora de crearlos.

5.2 SERVIDOR DE HERRAMIENTAS

5.2.1. Acceso a Objetos.

Los objetos descritos anteriormente se almacenan dentro de un servidor ottols. Esta es la denominación que recibe el servidor encargado de proporcionar las herramientas necesarias para el funcionamiento de un módulo de ekon.
Dicho servidor puede estar dedicado a proporcionar las herramientas o también ser cliente de esas herramientas, denominado pc-cliente. Para el desarrollo del escritorio se ha decidido crear una configuración pc-cliente conectado a un servidor ottols encargado de proporcionar las herramientas de creación del Escritorio.

5.3 CLASE PERSONALIZACIÓN

5.3.1. Introducción.

Las clases de personalización son código en lenguaje Java con las bibliotecas de Karat que permiten la modificación de objetos de negocio, formularios, consultas base, mensajes, lista de sistema y variables globales.
Dichas clases de personalización engloban el total de las herramientas de karat como los controles que contiene un formulario. Por ejemplo el control caja html solo es codificable mediante el uso de una clase de personalización.
Las clases de personalización permiten que el programador – desarrollador expanda las posibilidades de karat.
Dichas clases forman un paquete que dan funcionalidad al conjunto del Producto permitiendo una mayor riqueza en recursos de programación.
Por norma general un objeto de negocio o un formulario tienen asignada una clase de personalización.

5.3.2. Contenido, conceptos previos.

Tal como se ha comentado antes, las herramientas se alojan en un servidor externo o la propia máquina que solicita esas herramientas, por lo tanto, lo que controlan las clases de personalización son eventos que lanza el cliente, por ejemplo pulsar un botón, un calendario, cargar un formulario, cerrar un formulario, etc. Todos estos eventos son recogidos por una serie de funciones definidas en Karat que lanzan estos eventos.

Dichos eventos tienen un nombre ya definido en una superclase (la clase principal en orden superior a nuestra propia clase de personalización que contiene la

nomenclatura de dichos eventos). Por lo tanto el programador – desarrollador sólo tiene que controlar dichos eventos y su forma de ejecutarse y crear las diferentes funciones que permitan conseguir el propósito final.

Todos los eventos se realizan bajo una sesión de usuario. Cuando nosotros ejecutamos una aplicación de ekon se crea una sesión en el servidor que en función de los parámetros de esa sesión el servidor es quien establece qué permisos tiene el usuario. Dichos parámetros de la sesión permiten visualizar más o menos opciones en los formularios, ejecutar cambios o modificar información.

5.3.3. Cuerpo de la clase de personalización.

Para desarrollar una clase de personalización disponemos de la herramienta Eclipse, un software de programación, que contiene un plugin o extensión para añadir los campos de un objeto de negocio y los componentes de un formulario. Los campos de un objeto de negocio se denominan ítems y los componentes de un formulario que podemos distinguir entre contenedores, vistas y ventanas. El programador es quien decide en todo momento que debe utilizar para conseguir su propósito.

Por lo tanto a la hora de crear una nueva clase tenemos que tener en cuenta qué componentes estamos modificando que queremos modificar ya que cualquier cambio afecta a la base de datos. El cuerpo o contenido de una clase de personalización podríamos definirla de esta forma.

Figura 13: Imagen de los componentes que forman una clase de personalización ordenados por prioridad.

Cabe señalar que toda clase de personalización permite modificar cualquier parte de las herramientas antes mencionadas.

FASE DE IMPLEMENTACIÓN

6.1 INTRODUCCIÓN

6.1.1. Aspectos previos a la codificación.

Tal como hemos comentado antes, a la hora de codificar, existen unas piezas comunes a cada uno de los formularios que se han diseñado anteriormente Por ejemplo los colaboradores asignados a un Gestor, la planificación que tiene asignado cada uno, las incidencias, peticiones, etc. Toda esta información que es compartida por más de un formulario se debe definir antes, para que a la hora de codificar y generar una ventana se pueda hacer de una forma mucha más sencilla.
Por lo tanto previamente definiremos cuales han sido los pasos para crean los objetos comunes a cada formulario, para dar una visión global del proceso de codificación.

El sistema de datos es común a toda la aplicación ya que todos los formularios acceden a un mismo conjunto de datos alojados en diferentes tablas. A continuación se muestra el diagrama de tablas y la relación entre ellas de dónde se obtiene y se inserta la información. En el diagrama se puede observar una parte de las tablas que pertenecen a Ágora, en concreto control de presencia y planificación. En este caso se utilizan muchas más como pueden ser las tablas del producto ekon Común y las de los otros dos módulos restantes gestión de viajes y gestión de notas de gastos.

Las tablas que se muestran a continuación representan el esqueleto principal de la aplicación ya que varios formularios utilizan estas tablas para guardar físicamente los datos que se están modificando y que más adelante otros formularios utilizarán esa información para representarla.

El sistema de tablas es un sistema de datos relacional puesto que permite implementar base de datos ya planificadas o diseñadas previamente. Permite establecer interconexiones (relaciones) entre los datos que están guardados en las tablas y a través de dichas conexiones relacionar los datos de varias tablas. En nuestro caso con un mismo dato podemos acceder a varias tablas para poder acceder a la información.

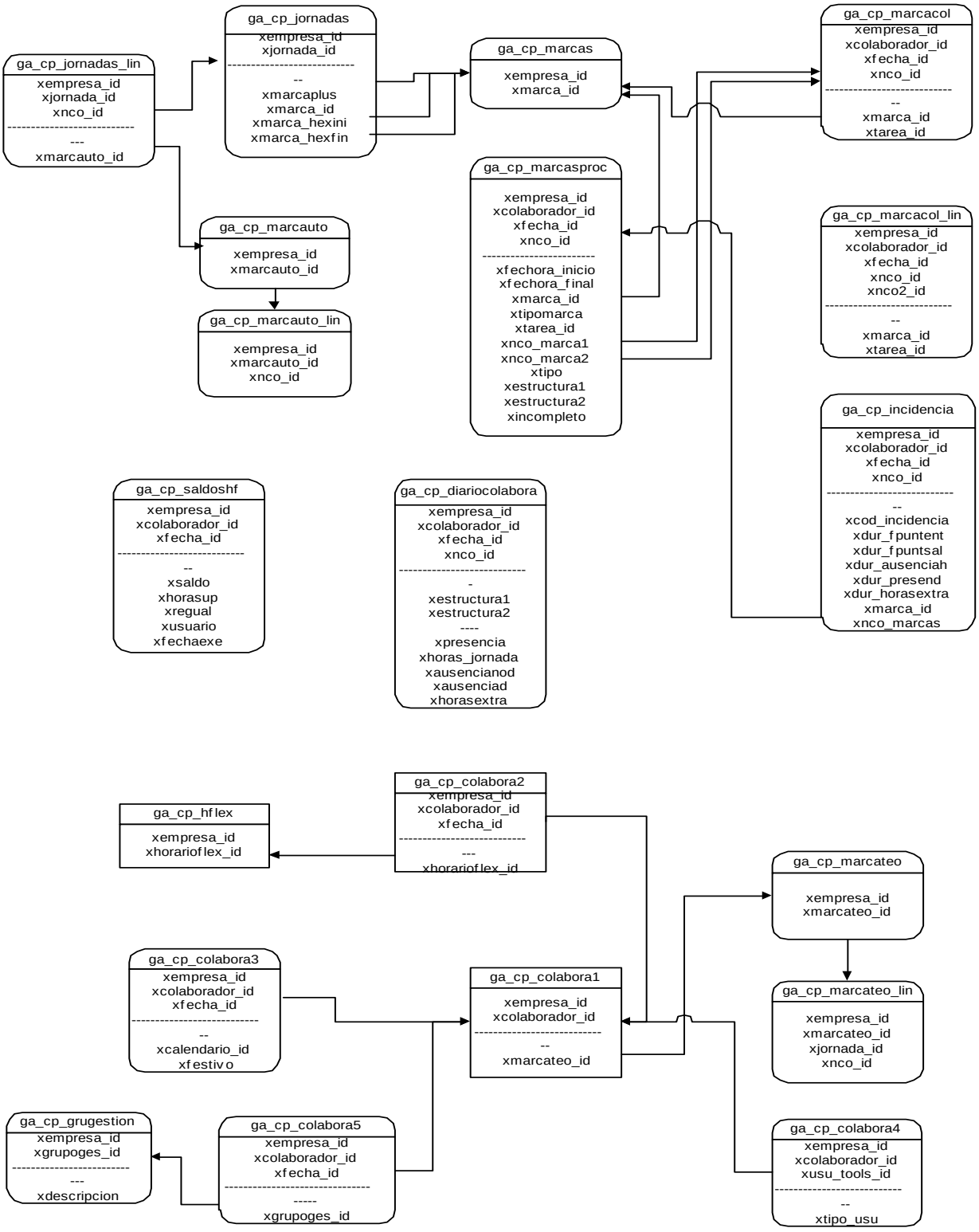


Figura 14: Relación entre las diferentes tablas más utilizadas para el funcionamiento de la aplicación.

6.2 OBJETOS COMUNES

6.2.1. Lista de colaboradores, código y nombre.

Los colaboradores en el entorno Ágora vienen definidos por varios valores: un código identificativo, su cargo u orden en la empresa, los privilegios, etc. en nuestro caso necesitamos el código interno de uso y el nombre que tiene asignado dicho código, además de la jerarquía que ocupa en dicha empresa. El nombre es el que mostramos por pantalla y el código será el dato representativo con el que trabajaremos a lo largo de todos los procesos para mostrar información en el escritorio.

Para crear una lista de colaboradores es necesario consultar la jerarquía de colaboradores de una empresa. La jerarquía de una empresa, en Ágora, se define mediante un árbol dónde la raíz es el gestor o responsable y las hojas de la raíz son los colaboradores dependientes del gestor. Por lo tanto hemos de guardar dicho árbol de dependencias en una lista para poder tener en orden cuales son los colaboradores que tiene el Gestor a su cargo. Dicha lista de colaboradores será necesario consultarla cada vez que se ejecute un procedimiento que necesite saber que colaboradores tenemos actualmente para procesar y mostrar información.

6.2.2. Planificación de un colaborador.

La planificación de un colaborador es uno de los factores más importantes en toda empresa. Se define el tipo de jornada que tiene un colaborador, hora de entrada, hora de salida, el horario, las incidencias que tiene planificadas, peticiones, vacaciones, días libres, etc. Por lo tanto esto es uno de los puntos de información que varios formularios necesitan para representar información a parte del formulario que representa gráficamente la planificación y el detalle de cada uno de los días. Comenzaremos por describir el proceso de creación de la ventana planificación ya que hay bastantes ventanas, en concreto tres, que necesitan dicha información.

6.3 VENTANA DE PLANIFICACIÓN

6.3.1. Objeto de negocio y campos que contiene.

Previamente debemos incluir en el objeto de negocio los campos en los cuales guardaremos el código del colaborador y su nombre, para poder guardar la lista de jerarquía de colaboradores que tiene el Gestor asignados.

En la descripción del proyecto la vista estándar de esta ventana contiene un grid o tabla, por lo tanto debemos definir los campos que contendrá entre grid para que puedan ser mostrados en el formulario. Por lo tanto creamos 31 campos dentro del objeto de negocio, que serán las columnas de nuestro grid, junto con los colaboradores. Todas las columnas las definiremos de tipo texto. Estos campos nos mostrarán la información horaria, semanal o mensual de los colaboradores en función de la selección de información. En este caso estos campos no tienen ningún tipo de información asociada, puesto que la información será insertada mediante la clase de personalización dependiendo de la opción que se desee visualizar.

La vista maximizada tal como hemos descrito anteriormente contiene dos filtros, fecha de inicio y fecha fin, para poder filtrar los días, meses o semanas que queremos visualizar, por ello creamos dos campos más, fecha inicio y fecha fin que serán de tipo fecha. El objeto de negocio creado se llamará ga_esc_planificacion. Todos los campos del objeto de negocio estarán vacios, es decir, no necesitarán que previamente se carguen los datos de una consulta base. Dichos campos se cargarán mediante código en la clase de personalización.

6.3.2. Formulario

Una vez tenemos el objeto de negocio realizado procedemos a la creación del formulario que mostrará los datos del objeto de negocio y añadirá la funcionalidad a la ventana. Por lo tanto en el formulario creamos las tres vistas, en la estándar, añadimos el grid con los 31 campos que hemos definido en el objeto de negocio, además del campo colaborador. Una vez creado añadimos el control calendario de karat, junto con los tres botones que darán funcionalidad al grid y una serie de cajas de texto, en total 6, que mostrarán información en función de lo que seleccionemos en el propio grid.

En la vista maximizada tenemos que añadir dos cajas html que permiten inyectar un código html para generar una determinada funcionalidad en html. En la primera insertamos los colaboradores y en la segunda la planificación de cada uno de ellos. Añadimos los controles de intervalo del objeto de negocio, un campo que mostrará el año que estamos visualizando y dos botones que nos darán la opción de poder visualizar por los parámetros insertados en los campos de fecha o poder visualizar por completo todo otra vez.

6.3.3. Clase de personalización, pasos previos

La creación de los elementos básicos que son puramente de karat, es decir, los controles que mostrarán información y darán usabilidad a la aplicación ya están creados, sólo falta dar funcionalidad a cada uno de ellos. Puesto que los controles son reutilizables, un mismo control tiene diferentes opciones de visualización y muestreo de la información y debemos controlar estas opciones mediante la clase de personalización.

Primero de todo debemos guardar la información en estructuras de datos auxiliares para que al cambiar de opción de visualización ya sea horaria, semanal o mensual no sea necesario volver a calcular otra vez la información de los colaboradores. Solo con acceder a las estructuras de datos auxiliares ya tenemos la información y de esa forma lo único que varía es la forma de representación de la información en función de la opción seleccionada.

La información de la planificación se almacena en una tabla denominada ga_pl_h_planing y que contiene, por cada día del año, que el colaborador tiene planificado, la sigla de su jornada (el símbolo y el color representativo de la sigla). La sigla y el símbolo no es lo mismo, la sigla se debe definir en todo momento a la hora que se define una jornada de trabajo y sólo puede haber una con ese nombre, al contrario que el símbolo ya que podemos tener varios con el mismo nombre. Hay muchas empresas que deciden no mostrar el carácter y si el color representativo, ya que una sigla sólo puede tener un único color. Esta tabla tiene asignada una consulta base con el mismo nombre que engloba todos los campos de la misma, cosa que facilita el acceso a la información.

Por otra parte, tal como se ha comentado, los colaboradores tienen una jerarquía asociada, y por lo tanto previamente debemos obtener la lista de jerarquía de la empresa y los colaboradores que tiene asignados el Gestor. Para ello sólo debemos crear un objeto de la clase de personalización de la primera ventana, presencia de colaboradores, y utilizar los métodos de esta clase que calcula dicha lista (en el aparatado de codificación correspondiente a esta ventana se explica el proceso) e insertarla en los campos del objeto de negocio que hemos definido previamente. De esta forma los campos que corresponden a colaborador y código de colaborador ya los tenemos informados. Estos campos serán inalterables durante toda la ejecución de la aplicación, sólo variarán el resto de campos del objeto de negocio.

6.3.4. Clase de personalización, guardar la información de la planificación

Antes de nada hay que tener en cuenta que karat calcula previamente los datos del objeto de negocio y los transmite al formulario, aunque también realiza cálculos previos a la hora de mostrar un formulario. Por lo tanto siguiendo con lo antes comentado, tenemos en un campo los colaboradores de los que tenemos que calcular su planificación, por lo tanto sólo debemos lanzar la consulta base asociada a la tabla de planificación por cada uno de los colaboradores que tenemos insertados en el objeto de negocio.
Para guardar los datos creamos doce vectores que contendrán tres filas y 30, 31 o 29 columnas respectivamente en función del mes con el que estemos accediendo.
Se deben controlar los casos en que un mes o días no estén planificados y por lo tanto esos días no deben contener información. La consulta que nos devuelve dicha información sólo devuelve los meses y los días que contienen información, por lo tanto el resto de campos no deben estar informados.

Una vez tenemos creados los vectores que contendrán la información, tenemos que guardarla por cada uno de los colaboradores. Por lo tanto debemos crear una lista para guardar la información. Esta lista contendrá los doce vectores creados anteriormente, que serán la planificación total de ese colaborador del año al que nos estamos refiriendo mediante el control del calendario que hemos añadido en la ventana. Añadimos los vectores que corresponden a cada mes a la lista de planificación del colaborador contengan o no información. Esta lista corresponderá a la planificación anual del colaborador.

Además si queremos tener la planificación ordenada por cada colaborador, debemos crear otra lista que contendrá la lista anterior, ya que hay que recordar que se irán añadiendo planificaciones en el mismo orden que vamos leyendo colaboradores. Por lo tanto tendremos en orden y en una misma lista la planificación de todos los colaboradores ordenada por meses y días.

Una vez finalizado el proceso, debemos incorporar las incidencias que tiene cada uno de los colaboradores añadidos en la planificación. Para ello debemos consultar y obtener qué días tienen planificado como incidencias y poner un símbolo descriptivo para ese día. Para ello utilizamos la consulta base ga_pl_h_inc_lin, que nos da los datos de la tabla con el mismo nombre con el cual se guardan los colaboradores y el año, mes y día junto con el tipo de incidencia que tiene, horas que comprende y comentario. En caso de encontrarse una incidencia añadimos un carácter a la sigla para determinar que en ese día hay una incidencia y debemos tratarla. (El carácter escogido es “**”).

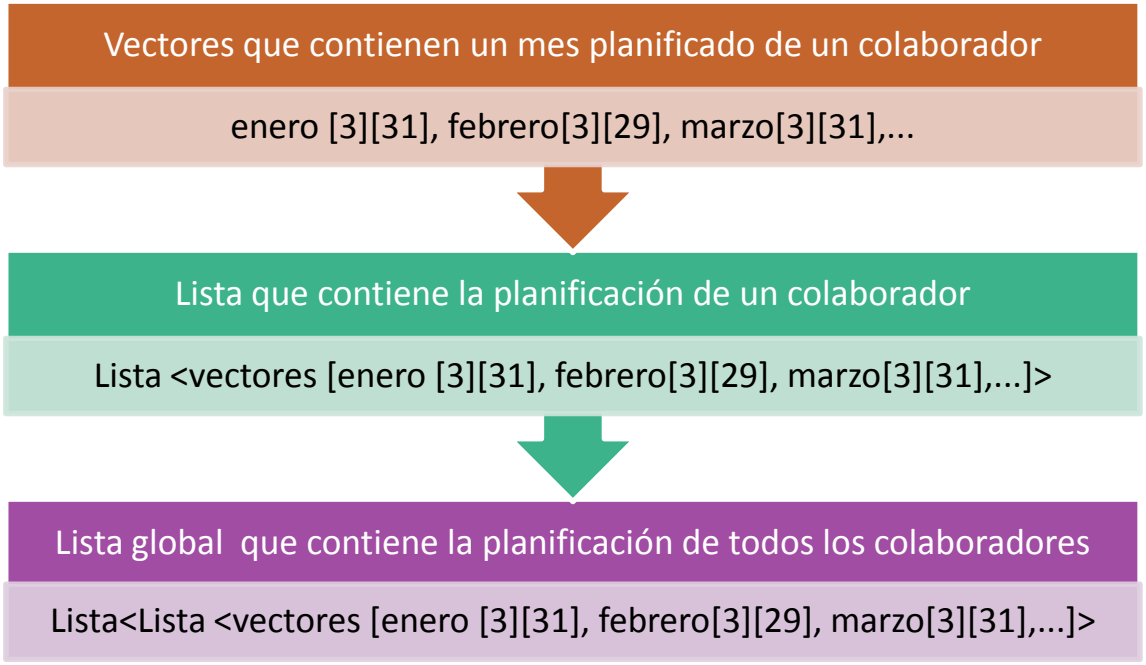


Figura 15: Diagrama de almacenaje de la información en listas de Java.

6.3.5. Representación de la información en función de la opción seleccionada, ventana estándar.

El proceso antes descrito lo realiza una función. Por lo tanto a la hora de ser utilizada por otras clases de personalización es tan sencillo como llamar a ese método de esa clase. Por defecto la aplicación, al cargar el formulario, muestra la información horaria del día actual, desde las 0:00 hasta las 24:00. El grid, en este caso sus columnas, muestran las horas. Cabe recordar que el grid tiene 32 columnas, por lo tanto desde la 24 hasta la 31 las hacemos invisibles. Para mostrar las horas en un grid que comprende una jornada de un colaborador en el día, mes y año actual, sólo tenemos que ir a la lista donde tenemos guardadas todas las planificaciones de todos los colaboradores y acceder en orden a cada una de ellas en función del día y el mes. Una vez tenemos la sigla que corresponde a la jornada debemos saber qué horas comprende para poder representarlas y si tiene algún tipo de incidencia.

Cuando obtenemos la sigla, si contiene "**" más la sigla, debemos ir a buscar qué horas tiene de incidencia y la jornada. De lo contrario sólo debemos ir a buscar la relación de horas. Las incidencias ya admitidas en la planificación se encuentran en la tabla ga_pl_h_inc_lin por lo que abrimos la consulta base, comparamos por fechas y si hay alguna en el día de hoy guardamos el tipo de descripción asociado, la sigla y las horas que comprende. Por último obtenemos la jornada que el colaborador tiene asociada. Que sería la capa por debajo de las incidencias. Dicha jornada se encuentra

en la consulta base ga_cp_jornadas_lin y accedemos a ellas como referencia de la sigla y obtenemos descripción, color representativo y las horas que comprende.

Una vez tenemos estos datos, sólo tenemos que tener en cuenta las horas de inicio y fin para ir representando el color de la jornada en cada una de las columnas del grid. Las columnas tienen relación de nombre con lo que representa, por ejemplo la columna 15 equivale a las 15 horas de un día y al campo 15 en el objeto de negocio. Así mismo se representan incidencias y horas de planificación, dando lugar a que si una incidencia afecta a una hora de jornada, la incidencia se representa por encima de la jornada. Por lo tanto sólo debemos ir insertando el carácter correspondiente a la sigla en los campos que corresponde en el objeto de negocio en la fila que corresponde al colaborador. Los colores que contiene cada celda se tienen que insertar al propio grid después del paso previo de insertar el carácter en el objeto de negocio. Realizado este paso, tenemos representado el grid de planificación horaria mediante los tramos horarios de cada colaborador y sólo falta controlar los casos de vista semana y mensual.

La funcionalidad del grid permite representar cualquier mes y semana de la fecha que seleccionamos en el calendario. Por lo tanto debemos crear un calendario auxiliar en una matriz de 6 filas y 7 columnas. Las filas corresponden al número máximo de semanas que puede contener un mes y las columnas a los siete días de la semana. Esta matriz permite saber en qué posición. (Lunes, martes, etc.) Le corresponde a un día determinado dentro de un mes seleccionado y poder insertar de esta forma el nombre de la cabecera de la columna. A la hora de traspasar la información a la vista semanal, a la hora de crear la matriz del calendario, guardamos en que semana se sitúa nuestro día seleccionado, es decir el día actual. De esta forma simplificaremos el cálculo ya que solo debemos hacer visibles las columnas que se corresponden con los días de la semana en la cual está situado el día seleccionado.

Una vez tenemos qué columnas se hacen visibles y cuáles no, sólo debemos pintar el grid y traspasar dicha información al objeto de negocio. Los campos del objeto de negocio se corresponden con los del grid, por lo tanto es relación 1 a 1.

En vista mensual y semanal sólo debemos volcar la información de la lista que contiene las planificaciones de los distintos colaboradores al objeto de negocio. Cada fila del objeto de negocio equivale a un colaborador, siguiendo el mismo procedimiento que la vista horaria pero sin tener en cuenta las franjas horarias de una jornada. En el caso de la vista semanas y mensual sólo se debe insertar la información que nos devuelve la lista de planificación.

Cabe señalar que a cada cambio producido en el calendario se debe borrar tanto el grid, el cálculo del calendario auxiliar y el objeto de negocio, ya que puede darse el caso que en esa selección de fechas no haya datos y se conserven los datos antiguos que no tienen relación con los nuevos.

6.3.6. Representación de la información en planificación global, ventana maximizada.

La ventana maximizada muestra el total de la planificación anual de todos los colaboradores en el control html box (caja html). Se decidió usar este control y no un grid como en la vista anterior ya que el uso de recursos de red es mucho mayor en el grid que en una caja html, puesto que la caja html se envía por la red una sola vez y al contrario del grid donde se tiene que enviar cada una de las columnas, en este caso 365 columnas por cada uno de los colaboradores.

La particularidad que tiene esta ventana es que se tiene que generar un código html por cada colaborador. Por lo tanto el método utilizado es añadir una cabecera que contenga todos los días del año (incluido si es bisiesto) y por cada colaborador creamos una línea con el total de su planificación. Sólo tenemos que ir transcribiendo esta información a código html. Creamos una tabla dónde cada día será una celda y su planificación será una fila. Por lo tanto creamos un bucle que extraiga dicha información de la lista y nos cree el código html asociado. Seguiremos el mismo proceso para conseguir la lista de colaboradores, sólo debemos acceder a los campos que contienen el nombre de colaborador y traducirlo a código html.
Por último, al disponer de filtros para visualizar la información, en caso de aplicarse por parte del Gestor lo único que debemos hacer es construir las filas en función de los meses y días que comprenden los filtros. El año depende del año seleccionado en el calendario de la ventana estándar.

6.3.7. Mostrar información del día seleccionado en el grid de la ventana y en control html.

Una de las funcionalidades que tiene el grid es poder seleccionar un día en las diferentes vistas y ver el detalle de las siglas o de las incidencias. En estos casos debemos ir a la lista de las planificaciones, saber la fila del grid seleccionada y la columna, en la vista horaria necesitamos el día del calendario, y acceder a los datos del colaborador que hemos seleccionado en el grid. Buscamos los datos sobre incidencias, si tiene, y jornada y representamos la información. A la hora de crear el código html sólo debemos crear un hipervínculo a cada celda, que al seleccionar esta celda nos devuelva la sigla que contiene dicha celda, de eta sólo se debe repetir el

paso antes descrito al seleccionar una determinado día en el grid de la ventana estándar.

6.4 VENTANA DE INCIDENCIAS

6.4.1. Objeto de negocio

En el caso de esta ventana, se decidió partir de la base del actual formulario de Karat para una mayor funcionalidad y visualización. La información que muestra el actual formulario es muy completa, pero no está agrupada, además de no poder filtrar los resultados. Por lo tanto en este caso, para no interferir en el objeto de negocio original y en el formulario, se decidió hacer una copia y trabajar sobre el mismo y utilizar la misma clase personalización.

El objeto de negocio y el formulario en cuestión son los ga_cp_inc_wi donde todos los parámetros que se utilizan son los que nos han de servir para rediseñar y readaptar las nuevas funcionalidades.
Dicho formulario se podría haber creado desde 0, pero teniendo en cuenta que el resultado esperado seria el mismo que el que ya hay creado, se decidió readaptarlo y dotarlo de una mayor funcionalidad.

6.4.2. Formulario

El anterior formulario permitía al usuario poder insertar un código y de esta forma poder ver las incidencias que tiene el colaborador introducido en el campo del formulario correspondiente. Por lo que se necesitaba una interacción con el formulario para ver la información, cosa que se desea evitar a la hora de crear un escritorio. Al mismo tiempo sólo permite la visualización de los datos de un solo colaborador, por lo tanto si el Gestor desea ver los de todos sus colaboradores la herramienta no lo permite.

6.4.3. Representación de la información, ventana estándar.

En la ventana estándar necesitamos representar la información por colaborador, tipo de incidencia que tiene asociada y número de registros contiene, de esta forma el Gestor puede visualizar de una forma ordenada la lista de incidencias. Para ello, en el objeto, de negocio creamos los campos que nos permitirán añadir dicha funcionalidad: el colaborador, el código, el tipo de incidencia y la cantidad.

En los campos del objeto de negocio colaborador y código de colaborador guardamos los nombres de los colaboradores y sus códigos asociados que nos facilita la lista de jerarquía de la clase de personalización de la ventana presencia. El tipo incidencia y la cantidad los insertaremos a partir del grid general que contiene las incidencias.

Para que el grid general muestre todas las incidencias de todos los colaboradores asociados debemos modificar el código de la clave de personalización y añadir una parte que al cargar el formulario coja los datos del usuario que ha iniciado sesión en el servidor, en este caso el Gestor. De esta forma conseguimos cargar los datos desde el principio. El grid general pertenece a la ventana maximizada, como ambas se cargan a la vez en lo que se refiere a datos, podemos utilizar dicha funcionalidad para mostrar los datos en la ventana estándar.

Leemos el grid y, por cada línea de incidencia, leemos el colaborador y el tipo de incidencia. Por lo tanto tendremos en la lista todas las incidencias de todos los colaboradores, ahora sólo tenemos que ir recorriendo la lista y por cada colaborador mirar la incidencia, el tipo e ir contando cuantas veces se repite. De esta forma obtenemos el número de veces que se repite una incidencia, por tipo y así tenerlas agrupadas.

Una vez tenemos los tipos de incidencia, que es un valor numérico, necesitamos su descripción. Por ello debemos ir a una lista de sistema, la sya_cp_incidencia, que en función de ese número nos devuelve qué tipo de incidencia es. Transcribimos la información obtenida al Grid y obtenemos la información de la ventana estándar. Estos pasos se deben definir en la clase de personalización.

6.4.4. Representación de la información, ventana maximizada.

Tal como se ha descrito antes, la ventana maximizada carga previamente todos los datos. De esta forma sirven para la ventana maximizada como para la minimizada, además de mostrar los datos a la vez que se carga previamente la ventana, sin tener que interactuar con la aplicación.

Una de las principales funcionalidades aplicadas a esta ventana es la de poder visualizar todas las incidencias, cosa ya realizada, y además poder filtrarlas por colaborador. Para conseguir el filtro añadimos al formulario un control que permite una lista desplegable a la que añadimos los colaboradores del Gestor que tienen incidencias, definidos en la lista previa que hemos construido a la hora de generar la ventana estándar con su información. De esta forma, cuando el Gestor seleccione un colaborador del control lista desplegable, sólo tenemos que eliminar las líneas que

contengan el código del colaborador aunque en la lista sólo contendrá los nombres de los mismos. Para poder obtener el código del colaborador debemos buscar en el objeto de negocio la correspondencia entre nombre y código de colaborador. Para eliminar las filas del grid necesitamos guardar que filas son y de esta forma podemos borrarlas de una forma directa.

El resto de funcionalidades del formulario quedan inalteradas, cómo son los filtros del propio grid y la visualización del grid principal, ya definidos anteriormente en el apartado 5.2.5. Descripción del proyecto.
La ventana minimizada muestra el total de incidencias que el Gestor tiene por solucionar.

6.4.5. Gráfico descriptivo de los pasos para mostrar la información.

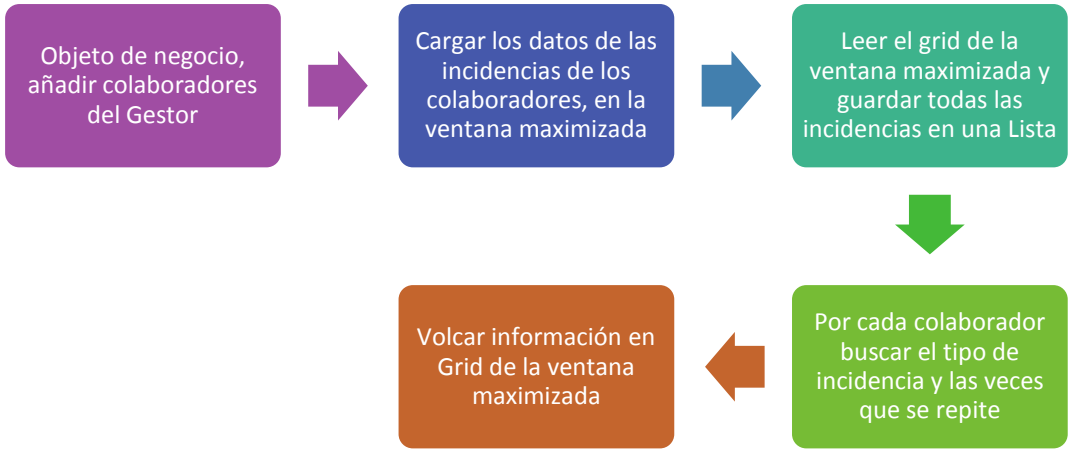


Figura 16: Diagrama descriptivo del proceso de incidencias.

6.5 VENTANA DE LÍMITES

6.5.1. Diseño del objeto de negocio.

La ventana de límites se ha creado partiendo de cero, puesto que el actual formulario sólo mostraba información de un solo colaborador y con poco detalle.

En el objeto de negocio se han añadido los datos colaborador y código del colaborador, ya comunes a todos los formularios a más de los siguientes campos necesarios: horas a cómputo, horas a servicio y horas de absentismo. Estos campos han sido duplicados para los dos tipos de vistas, la estándar y la maximizada. Dichas horas muestran las horas que ha realizado un colaborador o tiene asignadas y las horas de disfrute. Añadimos los tres campos que contendrá cada colaborador tanto

en la vista maximizada como en la estándar. Para rellenar dichos campos utilizaremos la consulta base ga_pl_h_consulta que nos calcula estos tres campos referentes a las horas.

6.5.2. Diseño del formulario, ventana estándar.

En la ventana estándar se ha insertado un grid en el cual el Gestor puede observar por cada colaborador las horas de cómputo, de servicio y de absentismo. En este caso el cálculo se realiza en el objeto de negocio por lo tanto no debemos tener código asociado en la clase de personalización asociada a esta vista.

6.5.3. Diseño del formulario, ventana maximizada.

En la ventana maximizada intervienen los demás factores antes descritos, como son las horas de cada colaborador, los límites y el año al que se refieren. Un colaborador puede tener tantos límites como sean necesarios, pero por lo general suelen ser cuatro límites asociados a cada colaborador en un año. (Pero es una condición que debemos tener en cuenta).

En el formulario, a la hora del diseño, añadimos un campo año actual, que permite visualizar los límites de los años que necesitemos. Añadimos un grid donde mostraremos las horas de la vista estándar pero con la funcionalidad que la información cambia en función del año seleccionado. Por último se han añadido los campos que contendrán los límites, que serán cuatro filas formadas por cinco campos cada una donde se mostrarán los límites con su respectiva descripción y las unidades restantes asignadas a ese límite.

La idea es que el Gestor seleccione un colaborador del grid superior, y automáticamente los límites asociados a ese colaborador se carguen en el formulario. Cabe señalar que si tiene más de cuatro límites asociados se nos muestra un botón para ver los restantes. Por lo tanto el grid principal de esta ventana, al ser de información variable en los campos referentes a horas, debemos controlar el borrado de la información en el objeto de negocio cada vez que el colaborador cambie el año de visualización.

Para dar funcionalidad de selección a toda la línea, es decir, que el usuario no necesariamente tenga que seleccionar al colaborador para mostrar la información sino que al seleccionar la línea los campos de los límites ya se muestren inmediatamente, se guarda el código de colaborador en una lista y de esta forma el colaborador seleccionado tiene la misma posición en la lista que contiene su código.

De esta forma podemos lanzar la sentencia a la base de datos y obtener la información referente a los límites.

6.6 VENTANA DE PETICIONES

6.6.1. Diseño del objeto de negocio.

En esta ventana, debido a la complejidad de datos que mostraba, se decidió utilizar el mismo objeto de negocio, puesto que el cálculo de la información, los campos que utiliza, la representación de la información y estructura podría ser utilizado para el diseño del nuevo formulario. Se podría haber conseguido partiendo desde cero se decidió partir de la base del mismo objeto de negocio y poder así dotar de una mayor funcionalidad.

Lo que se busca es dejar la parte de obtención de datos inalterable y poder modificar el formulario que muestra la información además de añadir nueva información y parámetros al formulario existente, dotando de esta forma de mayores prestaciones al formulario y concretamente a la aplicación.

6.6.2. Diseño del formulario, ventana estándar.

Para que el usuario a simple vista pueda tener una visión global de las peticiones que tiene pendientes, se decidió añadir en la ventana estándar el tipo de peticiones desde las más inmediatas hasta las más antiguas. Para el Gestor es importante saber la fecha que solicita la petición de un determinado día, hora o intervalo de fecha, como podrían ser vacaciones, para tener en cuenta el margen de tiempo de actuación que tiene. Para ello utilizaremos una tabla temporal donde obtendremos todas las peticiones de todos los tipos, puesto que pueden ser permisos y vacaciones, tanto de horas o jornadas completa, cómo de marcas.

Como se trata de mostrar las incidencias más inmediatas, descartamos incorporar a nuestra tabla temporal las incidencias rechazadas y ya aceptadas. Sólo mostraremos las propias del Gestor y las del nivel inferior, es decir, las de los colaboradores que tiene a su cargo y que su vez tienen un colaborador en nivel superior que es quien valida dicha petición.
Para representar la información que está almacenada en el objeto de negocio sólo debemos ir guardando aquellas incidencias con las restricciones antes mencionadas e ir añadiéndolas a nuestra tabla de trabajo temporal. De esta forma una vez la tenemos rellenada, podemos traspasarla directamente al grid del formulario.

6.6.3. Diseño del formulario, ventana maximizada, opciones de visualización.

La ventana maximizada contiene diferentes filtros que permiten añadir peticiones a las ya mostradas o eliminar las representadas, además de diferentes opciones para su visualización que permiten al Gestor una mayor amplitud de visualización a la hora de tomar decisiones. Tenemos el muestreo de peticiones, el tipo de peticiones y último el intervalo de fechas. Para comprender la forma en que se muestra dicha información cabe comprender que sólo debemos calcular la información necesaria que el usuario el Gestor solicita.

Calcular todas las peticiones que un Gestor tiene pendientes, aceptadas, rechazadas, según el tipo, etc. es un cálculo demasiado complejo en lo que se refiere a tiempo de ejecución y acceso a datos. Cabe señalar que el tráfico de datos es demasiado extenso si necesitamos calcular a la vez todos los datos para que luego en un futuro esperemos que el Gestor solicite esa función en concreto.

Por lo tanto se ha diseñado un formulario en el que la primera vez sólo se muestran las peticiones más inmediatas, es decir, las que corresponden a permisos y vacaciones, y los colaboradores que tiene el propio Gestor. De esta forma el tiempo de cálculo y respuesta para obtener la información es mucho menor. Cada vez que un Gestor seleccione una opción de visualización sólo debemos añadir o eliminar al grid las filas que contienen información que se corresponde con los criterios de selección o que son dispares con el criterio de selección.

Al seleccionar una opción de visualización se ejecutan las consultas a la base de datos con las restricciones de cada opción. De esta forma sólo se calculan los datos que realmente se necesitan. Además en caso de deseleccionar una opción de visualización, previamente se guarda el número de línea que hemos insertado y de esta forma sólo hace falta borrar las líneas que hemos añadido previamente al deseleccionar una opción.

Con este sistema sólo calculamos los datos necesarios en cada momento de la interacción con la aplicación y no se sobrecarga el sistema calculando desde un inicio todos los datos de todas las opciones, puesto que puede ser que el usuario no crea conveniente utilizar dichas opciones.

Figura 17: Diagrama descriptivo del proceso de incidencias.

6.6.4. Diseño del formulario, ventana maximizada, opciones de petición.

Por último nos quedan las diferentes opciones de las que dispone el Gestor para poder visualizar una determinada petición. Tenemos las opciones de visualizar la petición en detalle y en planificación. Tanto en detalle como en planificación utilizamos los mismos parámetros, cómo son las fechas, los días que comprenden ambas y el motivo, la secuencia de personas que pueden autorizar dichas peticiones, entre otras. Para obtener ambas vistas en detalle de una misma petición, sólo debemos ir a la tabla donde guardamos el total de las peticiones y rellenar la información de una forma ordenada en los campos que se muestran en el formulario.

Como las opciones de visualización del formulario dependen de un botón, sólo debemos controlar que al seleccionar ese botón se envíen los parámetros que corresponden a la petición para poder rellenar los campos. En ambos casos es lo mismo, pues los campos son idénticos y muestran la misma información a diferencia de visualizar la petición en planing que aporta la opción de ver la planificación del propio colaborador que la ha pedido. Para visualizar dicha planificación no es necesario volver a crear las funciones que generan la planificación la ventana planificación, anteriormente descrita, ya nos calcula dicha planificación de los colaboradores. Por lo tanto sólo debemos crear un objeto de esa clase de personalización que calcule la planificación y de esta forma podremos utilizar dichas funciones.

Cuando creamos una clase podemos crear un objeto de ella. De esta forma todos los métodos que contiene esa clase pueden ser aplicados sobre ese objeto. Digamos que ese objeto es la herramienta que permite realizar los cálculos necesarios para poder obtener el resultado esperado. Cuando ya tenemos creado un nuevo objeto de la clase planificación sólo debemos indicar de qué colaborador queremos generar su planificación y el propio mes a obtener. De esta forma se calcula el mes planificado y se genera el código html que más tarde inyectaremos en el control htmlBox del control de la ventana de detalle. De esta forma nos ahorramos volver a crear nuevas funciones y métodos que conllevarían una sobrecarga de los recursos del sistema y un gasto innecesario de los mismos.

6.7 VENTANA DE FICHAJES Y MARCAS.

6.7.1. Diseño del objeto de negocio.

Para la funcionalidad de esta ventana se decidió de utilizar el objeto de negocio que disponía la aplicación, ya que permite realizar fichajes del propio colaborador, de colaboradores que tiene a su cargo, generar peticiones, etc. y añadir las nuevas funcionalidades al formulario además de eliminar las que no son necesarias.
Para mostrar los marcajes de un determinado colaborador necesitamos crear una tabla o grid dónde mostrar el conjunto de marcas que ha realizado en un día y la planificación que tenía asignada.
Para ello añadimos al objeto de negocio los campos: hora, tipo de marca y descripción de la marca. De esta forma podremos asociar los marcajes al objeto de negocio y que sean mostrados de una forma directa y rápida. También añadimos un campo que contendrá la fecha por la cual el colaborador podrá visualizar los datos.

Por lo tanto en este caso sólo hemos añadido los campos que necesitábamos para conseguir la representación de las marcas. Y no ha sido necesario modificar o eliminar ningún campo del mismo.

6.7.2. Diseño del formulario, ventana estándar.

Una de las principales características es que el formulario que estaba vinculado al objeto de negocio antes descrito era multipropósito, estaba diseñado para terminales táctiles, por lo cual incorporaba botones con iconos de grandes dimensiones y controles innecesarios para el propósito para el cual se desea esta ventana dentro del Escritorio. Por lo tanto los elementos que no se desean utilizar se han eliminado tal como teclados virtuales (táctiles), imágenes de la persona que estamos visualizando, el logotipo de la empresa, la descripción del propio terminal,

etc. Aquellos elementos que presentaban iconos grandes se han reconvertido, ya que no está permitido el uso de iconos grandes, y se han añadido nuevas funcionalidades.

De esta forma conseguimos que la ventana de fichajes tenga una medida reducida y con las opciones claras y definidas. La lista desplegable con los colaboradores que el Gestor tiene a su cargo substituye el sistema de introducir al colaborador mediante código descriptivo. Para conseguir esta lista sólo debemos traspasar los colaboradores del objeto de negocio al control que contiene la lista desplegable. Los datos del objeto de negocio los obtenemos de la clase de personalización que contiene la ventana presencia de colaboradores que nos facilita la lista de jerarquía. Para ello sólo tenemos que guardar en una lista los códigos de colaborador y en otra los nombres a los que corresponde cada colaborador. De esta forma cuando el colaborador seleccione un colaborador en dicha lista y la opción visualizar colaborador, solo debemos saber la posición dentro la lista desplegable ya que será la misma que ocupará en la lista de códigos de colaborador.

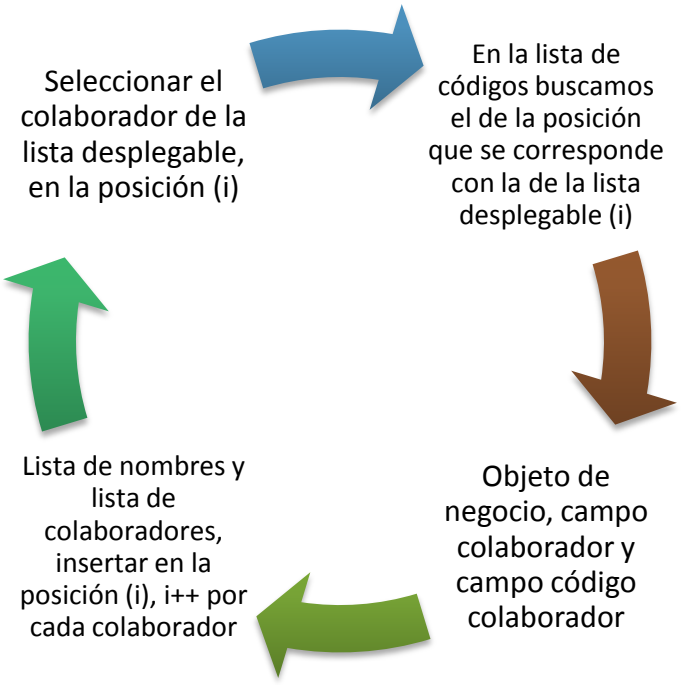


Figura 18: Diagrama descriptivo del proceso de incidencias.

El código del colaborador podemos asignárselo a una variable para poder operar toda la parte de fichajes. A la hora de realizar los fichajes de un Gestor y de sus colaboradores intervienen varios factores y varios procesos que son ajenos al desarrollo del Escritorio puesto que pertenecen a otros productos de ekon y que no intervienen en el desarrollo de la aplicación, ya que son independientes.

6.7.3. Diseño del formulario, ventana maximizada.

La ventana maximizada muestra el detalle de marcas de un colaborador que tiene asociado, la hora, el tipo y la descripción de la marca, además de la jornada que tiene y en caso de tener incidencias.
Todos estos campos dependen del colaborador seleccionado en la lista y de la fecha que queremos visualizar. Para obtener el código de un determinado colaborador seleccionado en la lista de colaboradores se realiza el mismo procedimiento definido en el anterior punto. El diagrama del proceso muestra los pasos concretos que se llevan a cabo.
Para la parte del detalle de jornada e incidencias del colaborador en función del día seleccionado se utiliza la clase de personalización de la ventana de planificación, de esta forma podemos obtener la jornada y las incidencias asociadas de un determinado colaborador, lo que permite que el Gestor visualice las marcas y la relación que conlleva a la jornada e incidencias.
Lo único que debemos crear es un objeto de esa clase y llamar a los métodos que nos permiten generar la planificación de un colaborador de un día y mes seleccionado.

Para generar la tabla de marcajes debemos ir a la tabla que contiene las marcas procesadas o admitidas de un determinado colaborador, en este caso el seleccionado, y de esta forma acceder a los registros que contiene para el día el cual solicitamos la información.
Como es de esperar, en dicha tabla, no se guarda la descripción de la marca, sino, un código asociado que indica la descripción de la propia marca. De esta forma, para poder rellenar la descripción de la marca sólo debemos ir a la lista de sistema que contiene dicha descripción en función del código introducido obtener la descripción guardada en la lista de sistema e insertarla en la fila del grid que corresponde.

6.8 VENTANA DE PRESENCIA.

6.8.1. Diseño del objeto de negocio.

Esta venta, a diferencia del resto, contiene un árbol de jerarquía que muestra los colaboradores que tiene asignado un Gestor. Para rellenar dicho árbol con sus respectivos nodos, no es necesario definir ningún campo en el objeto de negocio, puesto que no se utilizarán. Sólo debemos definir el control árbol de dependencias en el formulario.

6.8.2. Diseño del formulario.

En este caso sólo debemos añadir el control árbol de dependencias que más adelante, en la clase de personalización, añadirá las funcionalidades para representar la información en la ventana de presencia de colaboradores.

6.8.3. Clase de personalización, creación del árbol de jerarquía.

El árbol de jerarquía se obtiene a partir de diferentes consultas base en las cuales se combinan las diferentes jerarquías que pueden haber en una misma empresa, como pueden ser: autorizadores, validadores, generadores de petición, etc. Se pueden crear tantas como la empresa crea necesario. En nuestro caso siempre debe haber una, la de autorizadores que es la que debemos consultar y acotar para el Escritorio.

La jerarquía se obtiene mediante una consulta base que nos devuelve un nodo que es la raíz del árbol. Dicha raíz inicial es el colaborador superior, es decir, la persona que más cargo ostenta en esa empresa. En nuestro caso, como tenemos que mostrar el conjunto de colaboradores que tiene asignado un determinado Gestor, sólo debemos ir buscando qué nodo es el Gestor y una vez encontrado, expandir en profundidad y en amplitud el árbol hasta encontrar el conjunto de colaboradores que tiene a su cargo, los nodos hijos, y si éstos, a su vez, tienen colaboradores dependientes. De esta forma sólo se muestra el conjunto de colaboradores que el Gestor tiene a su cargo pues el resto de colaboradores es irrelevante.

6.8.4. Clase de personalización, marcas de los colaboradores.

Por cada colaborador, a la hora de generar el árbol de jerarquía, se determina si tiene marcas respecto al día actual. Por lo tanto cuando un nodo se convierte en raíz, es decir se está procesando para saber si tiene más dependencias asociadas, se calcula si tiene marcas procesadas asignadas al día actual.

En caso de tenerlas nos devuelve un código identificativo de la jornada, que más adelante se utilizará sobre una lista de sistema para poder obtener el símbolo que corresponde a la marca que tiene asociada el colaborador.

6.8.5. Clase de personalización, lista de jerarquía.

La lista de jerarquía es utilizada por todas aquellas ventanas que necesitan saber qué colaboradores están actualmente en orden inferior al Gestor. Dichas ventanas son:

planificación, límites y marcajes. La lista se obtiene de la misma forma que se crea la representación gráfica del árbol pero con funciones distintas, para que no sea necesario representar el árbol gráficamente todo el rato. Por lo tanto se ha creado una función con las mismas consultas base que la función que genera el árbol gráfico pero que como parámetro de entrada tiene una lista la cual rellena con los colaboradores en orden de posición jerárquica que tiene el colaborador asignado. De esta forma las clases de personalización que necesitan de esta función sólo deben crear un objeto de esa clase y llamar a esa función que obtiene los colaboradores ordenados jerárquicamente.

FASE DE PRUEBAS

7.1 Introducción.

El objetivo de las pruebas es determinar el correcto funcionamiento de cada uno de los componentes de la aplicación. De esta forma podemos determinar los fallos que se producen en cada una de las ventanas de forma más acotada. En nuestro caso las fases de prueba unitaria se han ido integrando a la vez que se finalizaba la fase de implementación de una ventana, tal como está descrito en la planificación del proyecto.
A continuación se detallan cada una de las pruebas que se han llevado a cabo en cada una de las ventanas y sus respectivos resultados.

7.2 Pruebas unitarias.

Las pruebas unitarias son aquellas que permiten asegurar que un determinado componente de la aplicación, en nuestro caso una ventana, devuelve una salida en correcta en función de una determinada entrada. En este caso nuestros parámetros de entrada son las acciones que realizan el Gestor y los parámetros de visualización de cada una de las ventanas y la salida es la información que solicita mediante dichas acciones. Modificando las acciones y los parámetros de entrada veremos que el resultado esperado sea el correcto. Para determinar si los resultados son los correctos en todos los casos se han contrastado con los valores que contiene la base de datos contra la cual se solicita, modifica o se inserta información.
Las diferentes pruebas que se han realizado dentro de esta categoría son las siguientes:

<u>Ventana de presencia de colaboradores:</u>

- Se ha comprobado que la jornada e incidencias junto con las horas que comprenden, corresponden a la jornada que tiene asignada en la tabla de planificación. Así mismo que el color de los iconos correspondan con los fichajes que ha realizado el colaborador. Ambos datos se han contrastado con las tablas que contienen dichos resultados.

<u>Ventana de incidencias:</u>

- Se ha comprobado que los filtros de las incidencias, tanto de parámetros como de colaboradores, funcionan correctamente. Solamente muestran las

incidencias indicadas en dichos filtros. La función de consultar día, si no hay ninguna seleccionada muestra la primera de la lista y que la información sea correcta respecto a ese día. Además, si el usuario valida algún cambio, este cambio queda reflejado en la que tabla contiene los datos de esa incidencia. La ventana minimizada tiene que mostrar el total de incidencias que tiene cada colaborador agrupadas por tipo, por lo tanto el total tiene que ser igual a la suma de las agrupaciones de incidencias. Para comprobar estos datos se han contrastado con las líneas de incidencias que contiene el colaborador en la base de datos.

Ventana de peticiones:

- Se ha comprobado que la ventana estándar muestre las peticiones de los colaboradores en orden en que se insertaron en el sistema.
  La ventana maximizada muestra las peticiones iniciales del Gestor que tiene por validar de sus colaboradores inmediatos según la jerarquía. En caso de querer visualizar las peticiones del nivel inferior se añaden correctamente junto con las que ya tenía en pantalla. El resto de filtros de visualización, además de los intervalos de fechas, muestran la información correctamente. En caso de no estar informados los campos de fecha, muestra un mensaje informativo.
  Por lo que respecta a la información de una petición tanto en planing como en detalle la información en ambas es idéntica y planing muestra la planificación asignada a ese mes. La información se ha contrastado con la base de datos y con los antiguos formularios de Ágora.

Ventana de planificación:

- En la ventana estándar se ha comprobado que inicialmente la ventana muestre la información horaria con las franjas horarias que comprende la jornada que tiene asignado un determinado colaborador. Además que al cambiar la vista a semanal o mensual la información sea la que tienen asignados a esa semana y mes. Por otra parte, al seleccionar un día en el calendario, la vista semanal, mensual y horaria, muestra la planificación asociada a la selección. Además al seleccionar un carácter en el grid de la vista, muestra la sigla correspondiente a esa sigla y las horas que comprende. En el caso que no haya carácter por decisión de la empresa se muestra sólo el color y el detalle horario de la sigla.
  En la ventana maximizada el año debe cambiar en función del calendario de la ventana estándar. Además los filtros del año, en caso de no estar informados, o ser incorrectos, muestran un mensaje informativo para su corrección. Al seleccionar una determinada celda de la caja html, el valor tiene que

corresponderse con el de la sigla seleccionada. Todos los datos se han comparado tanto con la base de datos como con los antiguos formularios de Ágora.

Ventana de límites:

- En la ventana estándar se ha comprobado que las horas de cómputo, servicio y de absentismo son las horas que tiene asignadas el colaborador. En la ventana estándar se ha comprobado que sean las mismas horas y que al seleccionar un colaborador del grid los límites que se muestran sean los mismos que el colaborador tiene asignados. Por otra parte, al cambiar de año, dichos límites cambian en función del año seleccionado. Todos estos datos se han contrastado con la base de datos.

Ventana de marcajes:

- En la ventana estándar se ha comprobado que se permite realizar marcajes del colaborador seleccionado, peticiones, consultas, modificar marcas y modificaciones.
  En la ventana maximizada se ha comprobado que una vez introducida una fecha y colaborador en la lista desplegable, el sistema muestra la jornada, las incidencias y los marcajes asociados a ese colaborador. Por último se controla el caso de que no se introduzca un colaborador y una fecha por parte del Gestor.

7.3 Pruebas de integración.

Las pruebas de integración tienen como finalidad probar el funcionamiento en conjunto de la aplicación. Esto quiere decir que las ventanas que realizan cambios en la base de datos y las que necesitan otras ventanas, estos cambios se realizan de forma correcta y conjunta, y que a la vez permitan realizar el proceso completo dando soporte a la aplicación. Los procesos verificados se detallan a continuación.

7.3.1 Alta y confirmación de nueva petición en la ventana marcajes.

En este conjunto de pruebas se verifica el alta y confirmación de una nueva petición en la ventana de marcajes.

- **Prueba 1: Alta de petición en la ventana de marcajes.**
  - Resultado esperado: Ver que la ventana de marcajes muestra un mensaje indicando que la petición asignada al usuario ha sido insertada correctamente.

- **Prueba 2: Seleccionar el colaborador en la ventana de marcajes y ver que tiene la petición asignada.**
    - Resultado esperado: Ver que al seleccionar otra vez el usuario en la ventana de marcajes y seleccionar la opción petición ya sale representada.

- **Prueba 3: Ver que la nueva petición es representada por la ventana de peticiones.**
    - Resultado esperado: Ver que la ventana de peticiones muestra la nueva petición con todos los parámetros insertados en la ventana de marcajes y con la opción pendiente de validar.

7.3.1 Validación de petición en la ventana peticiones y representación en la ventana planificación de colaboradores.

Este conjunto de pruebas verifica que al confirmar una petición en el sistema, la ventana de planificación de colaboradores la muestra correctamente.

- **Prueba 1: Validar una petición de un día completo sin jornada asignada a ese colaborador.**
    - Resultado esperado: Ver que en la ventana planificación en todas las vistas horaria, semanal, mensual y anual se ha insertado el carácter correspondiente a esa petición.

- **Prueba 2: Validar una petición de un día completo con jornada asignada a ese colaborador.**
    - Resultado esperado: Ver que en la ventana planificación en todas las vistas horaria, semanal, mensual y anual la jornada que el colaborador tiene asignada, tiene insertados dos asteriscos mostrando que ese día el colaborador tiene asignada una petición. Además en la vista horaria se muestra la franja horaria que ocupa dicha petición.

- **Prueba 3: Validar una petición de varios días con diferentes motivos y horas comprendidas.**
    - Resultado esperado: Ver que en la ventana planificación en todas las vistas horaria, semanal, mensual y anual la jornada que el colaborador tiene asignada, tiene insertados dos asteriscos mostrando que ese día el colaborador tiene asignada una petición. Además en la vista horaria

se muestra la franja horaria que ocupa dicha petición. En caso de no tener jornada asignada sólo se muestra la información de la sigla.

7.3.3 Validación de incidencia en la ventana incidencias y pasa a estar validada e insertada en el sistema.

Este conjunto de pruebas verifica que al confirmar una incidencia en el sistema, la ventana de incidencias ya no muestra la incidencia como pendiente y se ha insertado en el sistema para que otros productos de ekon que necesitan de las horas de incidencia puedan utilizarla.

- **Prueba 1: Validar una incidencia en la ventana de incidencias y que aparezca como validada en el sistema.**
  - Resultado esperado: Ver que en la ventana de incidencias la incidencia confirmada ha quedado procesada.

- **Prueba 2: Ver que el sistema tiene dicha petición como insertada.**
  - Resultado esperado: El sistema muestra la petición como insertada y procesada.

7.3.4 Actualización del estado de los colaboradores al realizarse una actualización de la ventana de presencia.

Este conjunto de pruebas verifica que al seleccionar la opción actualizar de la ventana de presencia de colaboradores se muestra el estado real del conjunto de colaboradores.

- **Prueba 1: Marcar la entrada de un colaborador.**
  - Resultado esperado: La ventana muestra un icono verde de entrada respecto a ese colaborador. El resto de colaboradores se muestran inalterados.

- **Prueba 2: Marcar la salida de un colaborador.**
  - Resultado esperado: La ventana muestra un icono rojo de salida respecto a ese colaborador. El resto de colaboradores se muestran inalterados.

- **Prueba 3: Cambio de iconos al realizar una marca.**
  - Resultado esperado: El icono inicial de presencia sin marcas asociadas cambia al realizarse cualquier marca por parte de ese colaborador.

Con estas pruebas basadas en un escenario de utilización se verifica que el conjunto de tablas que intervienen en todo el proceso se actualizan y que a su vez muestran la información de forma correcta. Se verifica que el Gestor puede realizar las acciones que permiten cada una de las ventanas de él mismo y de todos los usuarios que tiene a su cargo. Además se verifica que todos los campos tienen modificación física en el conjunto del sistema de datos.

CONCLUSIONES

8.1 Alcance de objetivos.

En este punto se hace balance de los objetivos conseguidos y del grado de cobertura de los objetivos fijados inicialmente en el desarrollo de la aplicación.

Podemos afirmar que el producto resultante de este proyecto cumple con los objetivos que se describieron al inicio del proyecto.

La aplicación desarrollada permite visualizar de forma conjunta los aspectos más importantes de los módulos control de presencia y planificación de Ágora. El usuario no tiene que abrir nuevas ventanas o nuevos formularios ya que la información más relevante y que requiere una mayor atención se muestra de forma inmediata y simple. Además, gracias a que las ventanas están interrelacionadas, cualquier cambio que se produce en el sistema es rápidamente reconocible por parte del Gestor y tiene el conjunto de colaboradores a su cargo agrupados por jerarquía. De esta forma es más fácil tener una visión conjunta de todos los eventos que se generan por parte de los colaboradores.

Estas funcionalidades hacen que se simplifique el trabajo del Gestor puesto que toda la información la tiene agrupada en único punto y en tiempo real. A su vez se agilizan los trámites a la hora de validar o rechazar el conjunto de peticiones, incidencias, etc. que realizan el conjunto de colaboradores.

El desarrollo de la aplicación no se limita solamente a la visualización de la información de forma conjunta, que era unos de los requisitos de la aplicación a obtener, sino que además permite al Gestor poder realizar cambios en las diferentes ventanas sin tener que abrir nuevas aplicaciones. Estas funcionalidades abarcan el total de las funcionalidades descritas en el inicio del proyecto.

Finalmente cabe destacar que el proyecto de fin de carrera se ha desarrollado siguiendo la normativa aplicable a su realización juntamente con la normativa de UNIT4.

8.2 Ampliaciones y mejoras.

A pesar de que la aplicación obtenida cubre los objetivos principales del proyecto, se puede enumerar un conjunto de mejoras o nuevas funcionalidades que no se han desarrollado.

La aplicación actual está basada en un modelo de empresa ficticio y por lo tanto el Gestor está indicado mediante código para que la aplicación utilice a dicho colaborador como Gestor. Este punto requiere que cuando un usuario se identifique al iniciar sesión se compruebe si es un Gestor y se carguen los datos referentes a él.

Actualmente las ventanas utilizan su propio calendario para poder seleccionar la información. Está contemplado que dichas ventanas obtengan los datos de fecha mediante un calendario global del propio escritorio, común a todas las ventanas que lo soliciten.

Tal como se ha comentado anteriormente los aspectos más relevantes de los módulos de control de presencia y planificación han sido implementados, pero, puesto que la herramienta escritorio está en fase de desarrollo, en un futuro se podrán insertar nuevas funcionalidades a cada una de las ventanas.
Por último destacar que esta fase corresponde a la creación del primer escritorio gestor, con la información más relevante de Ágora. Por lo tanto en un futuro se podrán añadir nuevas ventanas que contengan otro tipo de información y nuevas funcionalidades.

8.3 Desviaciones respecto la planificación inicial.

Aunque que las fechas de finalización del proyecto se han cumplido, cabe señalar que muchas fases de creación de ventanas se han visto incrementadas en la recta final del desarrollo del proyecto. Con esto nos referimos a que la ventana de control de presencia, aunque fue la primera en crearse inicialmente con sus respectivos test y pruebas se decidió en las fases finales del proyecto crearla de nuevo e incorporar nuevas funcionalidades.

Por otra parte un requerimiento añadido al proyecto ha sido que el diseño de todos los nuevos formularios debía seguir el estilo walnut cosa que ha impedido la creación de un manual de usuario y por lo que se ha desestimado su creación. En su lugar se creará un manual de cómo está confeccionada la aplicación.

8.4 Valoración personal.

Referente al ámbito académico la confección de este proyecto ha supuesto la creación de una nueva aplicación desde cero siguiendo todos los pasos necesarios para poder desarrollarla. A su vez ha permitido adquirir conocimientos en nuevos lenguajes de programación y enfrentarse a nuevos retos de programación y diseño.
Además ha sido necesario documentarse sobre los diferentes módulos que ya existían así como las páginas de código que los componen suponiendo un valor añadido a la hora de confeccionar la aplicación.

A nivel personal, poder desarrollar el proyecto en una empresa de software de ámbito mundial ha supuesto un valor añadido a la realización del proyecto puesto que el entorno de trabajo ha sido el que me encontraré en mundo laboral.
Además supone la finalización de mis estudios de ingeniería técnica informática y supone un gran paso en la culminación de mis estudios.

Finalmente dar las gracias a todas las personas tanto de UNIT4, de la UAB y a mis compañeros que han hecho posible la creación de este proyecto.

BIBLIOGRAFÍA

A continuación se detallan las referencias bibliográficas consultadas para el desarrollo del proyecto. En este caso se han utilizado tanto referencias Web como documentación de Ágora que pertenece a UNIT4.

Documentos de UNIT4, Ágora

- Ficha técnica de Ágora.
- Manual temático.
- Módulos y descripción que componen Ágora.
- API de karat

Documentos electrónicos.

- http://www.programatium.com/biblioteca/diseno/codigosdecolor.hth
    - Web que explica la relación entre los colores RGB y HTML.
    - Fecha último acceso 20/05/2011.

- http://lineadecodigo.com/java/obtener-fecha-actual-con-java
    - Web que explica el manejo de las fechas en Java.
    - Fecha del último acceso 26/05/2011.

- http://todojava.awardspace.com/
    - Web con ejemplos de código de Java.
    - Fecha del último acceso 30/05/2011.

- http://es.wikipedia.org/wiki/Wikipedia:Portada
    - Enciclopedia online.
    - Fecha del último acceso 30/05/2011.

- http://www.google.es
    - Buscador online
    - Fecha del último acceso 10/06/2011.

CONTENIDO DEL CD.

- Memoria del Proyecto en formato PDF.
- Repositorio del producto ekon Ágora, incluye modelo de datos y los objetos de karat utilizados en el proyecto.
- Clases de personalización Java de los objetos de karat.
- Presentaciones realizadas en UNIT4.
- Manual temático de ekon Ágora.
- Manual técnico de ekon Ágora.

Lunes 28 de Junio de 2011
Pablo Máximo Elorga Martín