SoftBank

Memoria del proyecto

Miembros del equipo:

Ignacio García Sánchez-Migallón	Diego Sánchez Muniesa
David Gorricho San Juan	Iván Monterrubio Cerezo
Carlos Gavidia Ortiz	Lorenzo José de la Paz Suárez

Índice

1. Estructura del proyecto

1.1 Diagramas usados

Los diagramas que hemos usado a la hora para poder implantar nuestro proyecto han sido:

- **Diagramas de casos de uso:** En cada diagrama se hace referencia a cada caso de uso, que dispone nuestra aplicación, incluyéndose lógicamente los actores que intervienen en cada caso de uso.
  Muestran las relaciones entre los diferentes actores y cada caso de uso, que describimos en la Especificación de Requisitos y que no hemos modificado.
- **Diagrama de actividades**: Muestra las actividades y como se relacionan entre ellas.
  Dicho flujo de eventos que se van a realizar en la aplicación lo representamos con dicho diagrama.
- **Diagramas de clases**: Va a mostrar el conjunto de clases, de las interfaces y de sus relaciones.
  Donde aparecen los distintos módulos de la aplicación que se van a realizar, los atributos y la forma en la que se relacionan entre sí.
- **Diagramas de secuencia**: En dicho diagrama muestra la secuencia explícita de mensajes y como sería la ordenación temporal de todos ellos.
  Este diagrama es uno de los que mejor explican las especificaciones a tiempo real y con nuestros escenarios.
- **Diagrama de despliegue**: Muestra la configuración de los nodos que participan en la ejecución y los componentes que tienen.
  Describe así de manera general los nodos de ejecución del programa.

1.2 Organización de los diagramas

Los hemos organizado en tres paquetes que son:

- **Modelo de requisitos**: El cual contiene un diagrama de clases de todos los módulos, también tendrá un paquete de actores que intervienen en los casos de uso. Por lo tanto, habrá 7 módulos en total y cada uno tendrá su diagrama de casos de uso en concreto.
- **Modelo de diseño**: El modelo de diseño estará compuesto por tres capas, integración, negocio y presentación. Y en cada capa tendrán los módulos correspondientes, en presentación y negocio tendrá los 7, pero en integración solo 1, acceso, ya que es el único que interactúa con el archivo txt, donde están los datos de nuestros clientes. En cada una de las capas descritas anteriormente tendrán tanto sus diagramas de clases como de secuencias, descritas en cada módulo.
- **Modelo de despliegue**: Está compuesto por el diagrama de despliegue de la aplicación, así como los entornos de ejecución necesarios para nuestro proyecto.

2. Diseño e implementación

El proyecto ha supuesto todo un reto para nuestro pequeño equipo, que creemos que hemos sabido afrontar y superar. En un principio, tal y como teníamos establecido en nuestra planificación temporal en Project, nos dividimos los diferentes módulos a desarrollar por parejas. Cada una de las parejas era la encargada de ponerse de acuerdo y desarrollar los diferentes módulos por su cuenta, desde su casa.

Sin embargo, pronto nos dimos cuenta que había algunas partes de módulos que, a lo mejor, una pareja no sabía desarrollar o no encontraba el enfoque adecuado que le tenía que dar y resultaba preciso pedir ayuda al resto del grupo para lograrlo. Por ello, a los pocos días de comenzar cambiamos nuestro método de trabajo. Seguimos trabajando por parejas cada uno en su módulo pero la principal diferencia fue el hecho de que estábamos todos en el mismo lugar físico. Es decir, en vez de trabajar cada uno desde su casa, quedábamos en uno de los laboratorios de la facultad y, cada pareja en un ordenador diferente trabajaba en su módulo aunque contando con la ayuda del resto del grupo. De esta manera, logramos aumentar la productividad y creemos que conseguimos unos mejores resultados.

Así, conseguimos, tras unos primeros días un poco ajetreados, que cada pareja pudiera realizar el análisis de cada módulo independientemente. Eran sesiones en las que, como ya teníamos la idea bastante clara, nos dedicábamos a perfilarla un poco más y a elaborar sus respectivos diagramas. Estos diagramas se realizaban siguiendo siempre la "guía de diseño" del equipo para así lograr tener, cuando juntamos todos los módulos, una aplicación en la que todos sus diagramas, diseño, código… fueran perfectamente homogéneos. Esto es algo que, de haber seguido trabajando con la metodología inicial, (cada pareja por su cuenta) no hubiera sido posible lograr o, de haberlo conseguido, hubiera resultado realmente difícil y complejo.

Posteriormente, tras tener la parte de análisis de todos los módulos completa, pasamos a tratar la parte de diseño. Podemos considerar que es la parte que más tiempo nos llevó ya que, creímos tenerla terminada en multitud de ocasiones pero nos dábamos cuenta al estar desarrollando código, que había parte del diseño que lo habíamos planificado mal. Tal era el caso de funciones que necesitaban más parámetros de los que tenían para poder llevarse a cabo, otras a las que les sobraban parámetros… Por eso, cada vez que nos dábamos cuenta que algo era incorrecto, debíamos ir a los diagramas, modificarlos y volver a generar las cabeceras para así trabajar con los nuevos métodos que sí incluían los parámetros correctos. Aunque al principio hemos de reconocer que nos costó mucho esta parte, al final le acabamos cogiendo el truco y es una auténtica maravilla ver cómo puedes modificar los diferentes modelos de diseño para así poder desarrollar el código como tú quieras.

Y por último (aunque ya hemos señalado en el párrafo anterior que no llegó a ser una diferencia muy clara entre la parte de diseño e implementación), pero no menos importante, nos tocó implementar el código. Hemos aprendido que hay una gran diferencia entre ponerte a hacer código a lo loco, sin tener idea de por dónde empezar, con cabeceras que vas modificando sobre la marcha, a planificar toda la aplicación bien, con sus respectivos diagramas de clase, de secuencia… y ver que, una vez llegas a la parte de la codificación, tan solo es rellenar los métodos con la información proporcionada en las cabeceras, a excepción de las pequeñas modificaciones que haya que hacer. No obstante, en esta última parte tuvimos un pequeño problema y es que, en un principio, nuestra planificación iba encaminada a tener el proyecto listo para finales de junio. Finalmente, dispusimos de unas tres semanas menos por lo que solamente nos ha dado tiempo a implementar dos módulos (si bien son los principales), el de acceso y el que permite gestionar la alta y baja de usuarios y empleados. Si hubiéramos dispuesto al menos de una semana más tal vez podríamos haber conseguido la implementación completa de la aplicación, aunque, a pesar de ello, estamos contentos con el resultado obtenido. Hemos implementado la aplicación haciendo uso de Java, ya que es un lenguaje muy dinámico, y que ofrece una gran cantidad de métodos que facilitan la creación de código.

3. Diagramas

3.1 Realización de los diagramas

Los diagramas del proyecto se encuentran repartidos en tres paquetes diferentes:

1. **Paquete de Modelo de Requisitos**
2. **Paquete de Modelo de Diseño**
3. **Paquete de Modelo de Despliegue**

El **paquete de modelo de requisitos** se encuentra compuesto por los diagramas del Modelo de Dominio, identificando las relaciones dadas entre los distintos (y numerosos) módulos del proyecto; un paquete con los distintos actores que representan a personas físicas que harán uso de la aplicación y por último un paquete por cada requisito funcional que incluye los diagramas de casos de uso.

El **paquete de modelo de diseño** se compone por tres paquetes diferentes, presentación, negocio e integración. En presentación se encuentran los diagramas de aquellos módulos del proyecto que se relacionan con la parte externa de la aplicación. En negocio se encuentran los diagramas de cada módulo reflejando la funcionalidad de los mismos y sus relaciones con las clases *Transfer*, por último en el paquete de integración se encuentran los diagramas de cada módulo que representan su relación información interna de la aplicación. Además, cada uno de los paquetes anteriores está compuesto, también, por sus respectivos diagramas de secuencia, mostrando la secuencia explícita de mensajes destacando su ordenación temporal.

Por último, el **paquete de Modelo de Despliegue** está compuesto por el diagrama de despliegue de cada módulo de la aplicación así como sus entornos de ejecución y objetos que lo componen, estos diagramas muestran la configuración de los nodos que participan en la ejecución de la aplicación.

3.2 Notación seguida en los diagramas

La notación seguida en la realización de los diagramas se ha escogido en base al patrón utilizado (Patrón Transferencia (***Transfer***), Patrón **D**ata **A**ccess **O**bject (**DAO**) y Patrón **S**ervicio de **A**plicación (***Application Service***).

Así, las clases de los diferentes diagramas tienen sus nombres basados en las reglas seguidas por los patrones utilizados (TransferUsuarioDAOVista, TransferUsuarioDAO, UsuarioDAO, EmpleadoDAO…)

4. Arquitectura

A lo largo de todo el proyecto, se ha utilizado la arquitectura multicapa, tanto sus estructuras como la lógica como las bases de esta misma. Todas ellas cuentas con patrones que sirven para identificar todas las clases, así como sus roles y comunicaciones proporcionando simplicidad y reusabilidad al diseño de nuestro proyecto. La arquitectura multicapa divide el proyecto en tres capas principales: una capa de presentación, otra de negocio, y otra de integración.

A estas tres, se le añaden la capa de clientes y la capa de recursos. Cada una de las capas contiene un conjunto de clases con responsabilidades relacionadas con la capa a la que pertenecen.

La capa de presentación es la encargada de encapsular la lógica del proyecto para que los usuarios puedan usar el sistema. Cuenta con dos patrones: El Controlador Frontal que proporciona un punto de acceso para el manejo de las peticiones de la capa de presentación y el Controlador de Aplicación.

La capa de negocio proporciona los servicios del sistema. El patrón más importante a destacar es el de transferencia (el llamado Transfer) para independizar el intercambio de datos entre las capas del proyecto. También cuenta con patrones como: Servicio de aplicación, patrón que sirve para centralizar simplificar y depurarla lógica de la capa de negocio u Objeto del Negocio.

La capa de integración es la responsable de la comunicación con recursos y sistemas externos a nuestra aplicación. El patrón en el que se basa esta capa es el **DAO** (**D**ata **A**ccess **O**bject), el cual nos permite acceder a la capa de datos externa, en nuestro caso, los '.txt'. Por ultimo cabe mencionar el patrón de Almacén del Dominio que separa la persistencia del modelo de los objetos (para modelos complejos).

La capa de clientes representa a todos los dispositivos o clientes del sistema que acceden al mismo.

La capa de recursos contiene los datos del negocio y recursos externos pero necesarios.

Es una arquitectura que está centrada en la reutilización y el mantenimiento, ya que se puede modificar cualquier capa sin afectar a las demás. También centraliza el control tanto de los accesos como de los recursos pero tiene inconvenientes como la sobrecarga o complejidad que pone en la red.

5. Patrones

Debido a la necesidad de describir la relación entre clases y objetos de una manera más intuitiva y factible, hemos usado patrones de diseño, tanto los patrones de propósito general, también llamados Patrones auxiliares, como los Patrones orientados a objetos.

5.1 Patrones usados

Los patrones usados en nuestro proyecto son los siguientes:

- Patrón **Modelo-Vista-Controlador** (***MVC***): usado como propósito general en la estructura de diseño, dividiendo nuestra aplicación interactiva en tres componentes: Modelo, el cual contiene la funcionalidad básica y almacena los datos; el Controlador, intermedia la comunicación entre la Vista y el Modelo, y a Vista, mostrando la información al usuario.

- Patrón **Transferencia** (***Transfer***): usado para independizar el intercambio de datos entre las capas de Integración, Negocio y Presentación.

- Patrón **Data Access Object** (***DAO***): usado para acceder a la capa de datos. Lleva el manejo total de los datos y su almacenamiento.

- Patrón **Servicio de Aplicación** (***Application Service***): usado para encapsular lógica no representada por objetos del negocio.

5.2 Organización de los patrones

El patrón **Modelo-Vista-Controlador** lo hemos utilizado de manera general en la aplicación, como se muestra en los diagramas de clase del paquete de diseño. Al dividirse en Integración, Negocio y Presentación los diagramas, empezamos ya a mostrar la separación entre la vista, qué es la presentación, el Controlador, que se encuentra dentro del paquete Negocio, y el modelo, que interactúa con los datos. Como se refleja en los diagramas de clase, tenemos un controlador (UsuarioControlador) que lleva toda la función de establecer la relación entre la vista y el modelo, aunque en la mayoría de los diagramas de clase de los módulos dejamos la función de comunicar con el modelo a los Servicios de Aplicación.

El patrón ***Transfer*** lo hemos usado para el traspaso de información entre las diferentes capas de la arquitectura. Los *transfer* creados en los diagramas son 2: TransferUsuarioDAOVista, que como su nombre indica pasa la información de la capa de presentación a la de negocio y TransferUsuarioDAO, que transmite la información entre las capas de negocio e integración.

El patrón **DAO** se asemeja a las clases UsuarioDAO y EmpleadoSA, las cuáles son interfaces con las operaciones CRUD como mínimo. Mediante dos clases que implementan estos DAO, realizamos la implementación de las operaciones, usando como recurso los ficheros de datos.

El patrón **Servicio de Aplicación** se ve muy reflejado en las clases UsuarioSA y EmpleadoSA, que contienen toda la lógica e información de los usuarios y empleados, de la aplicación.

6. Otros aspectos

En relación a la descripción y explicación de nuestro proyecto no habría muchos más aspectos que añadir.

La forma en que hemos organizado el desarrollo del proyecto ha sido la indicada por el plan de proyecto realizado con anterioridad. Se han intentado seguir los pasos que este nos marcaba, completando una a una las iteraciones de los distintos módulos, en la medida que nos ha sido posible por la restricción de tiempo ante la que nos presentábamos. Como se indica más arriba, hemos logrado completar las fases de análisis y diseño de los 7 módulos que forman nuestro proyecto, no de la misma manera que la implementación, que únicamente hemos finalizado en los módulos Acceso y Alta/Baja Cuenta, debido a falta de tiempo.

Un aspecto a destacar viene precisamente relacionado con el tiempo. Nos hemos visto obligados a cambiar en algunos puntos el plan de proyecto por el que nos guiábamos. El cambio más significativo es que hemos suprimido las tareas de codificación y pruebas de los 5 módulos en los que no nos ha sido posible producir el código, además de alguna pequeña modificación para ajustarnos al marco de tiempo que teníamos hasta la entrega final.

Algo que nos parece relevante indicar es la cantidad de veces que hemos tenido que reestructurar la fase de diseño al realizar los diferentes diagramas para solventar todos los inconvenientes que nos iban surgiendo, sobre todo a la hora de generar el código a partir de los diagramas. En ocasiones, observábamos que algún detalle no se adaptaba a nuestra idea original y nos veíamos obligados a cambiar algo en los diagramas. Hemos comprobado que merece la pena emplear tiempo en organizar y asegurarse de que todo está perfectamente planificado para evitar tener que retroceder y empezar a hacer las cosas, en algunas ocasiones, desde cero