

An abstract graphic consisting of several thin, parallel white lines that originate from the right edge and extend diagonally towards the top-left corner of the page.

SOFTBANK

SOFTBANK

Aplicación bancaria para empleados y clientes

Ignacio Pineda Martín

Índice

1. Memoria del proyecto.....	3
2. Objetivos del proyecto.....	4
3. Módulos formativos aplicados en el TFG	5
3.1 Programación.....	5
3.2 Desarrollo de interfaces	5
3.3 Base de datos	5
3.4 Acceso a datos	5
3.5 Entornos de desarrollo	5
4. Herramientas y Lenguajes utilizados.....	6
4.1 Eclipse	6
4.2 Apache NetBeans	6
4.3 Java EE	6
4.4 phpMyAdmin	7
4.5 XAMPP	8
4.6 GitHub	8
4.7 Diagrams.net.....	9
4.8 JDateChooser y JCalendar.....	9
4.9 iText 7.....	9
4.10 Java Persistence API (JPA)	9
5. Fases del proyecto	11
5.1 Idea	11
5.2 Primera aproximación.....	11
5.3 Modelo de datos	11
5.4 Base de datos	12
5.5 Modelo, Vista y Controlador (MVC)	13
5.5.1 Modelo.....	14
5.5.1.1 Modelo Entity	14
5.5.1.2 Modelo DAO	14
5.5.2 Vista.....	15
5.5.3 Controlador.....	15
5.6 Java	16
5.7 Swing	16
5.8 Diseño de Interfaces	17

5.9 Diagramas de clases.....	19
5.10 Validaciones relacionadas con el entorno bancario.....	20
5.10.1 CCC (Código de Cuenta de Cliente)	20
5.10.2 IBAN (International Bank Account Number)	21
5.10.3 PAN (Personal Account Number)	21
5.10.4 TAE (Tasa Anual Equivalente)	22
5.11 Validación de documentos de identificación.....	23
5.11.1 DNI.....	23
5.11.2 CIF	23
5.11.3 NIE	23
5.12 Exportación a PDF	23
6. Conclusiones y mejoras del proyecto	27
7. Bibliografía.....	28
8. Anexos.....	29

1. Memoria del proyecto

Este proyecto de fin de ciclo de Desarrollo de aplicaciones multiplataforma, es una aplicación de gestión y visionado de ciertas funcionalidades de un banco. La aplicación desarrollada implementa funcionalidades específicas de un supuesto empleado del banco, así como funcionalidades típicas de un supuesto cliente bancario.

La elección del tema de este proyecto se ha basado en varias premisas. La primera es que el tema permitía una gran flexibilidad a la hora de escalar el proyecto, pudiendo añadir nuevas características en función del tiempo disponible para el desarrollo del mismo. La segunda premisa es que el desarrollo de aplicaciones financieras tiene, en opinión del alumno, una alta aplicación laboral en el sector de la informática. La tercera y última premisa, es que esta aplicación permitía demostrar los conocimientos que el alumno ha adquirido durante el ciclo formativo.

En resumen, esta aplicación es un demostrador tecnológico, sin aplicación comercial, que ha permitido utilizar y ampliar los conocimientos adquiridos durante la formación, y conocer al mismo tiempo ciertos aspectos del ámbito financiero.

2. Objetivos del proyecto

En el proyecto desarrollado se han implementado las siguientes funcionalidades:

1. Una aplicación bancaria enfocada al empleado, que le permite lo siguiente:
 - a. Gestión de las sucursales y sus direcciones de la entidad bancaria.
 - b. Gestión de los empleados que forman la entidad bancaria.
 - c. Gestión de los clientes y sus direcciones de la entidad.
 - d. Gestión de los diferentes productos bancarios: cuentas corrientes, tarjetas de crédito y préstamos; incluyendo sus movimientos.
 - e. Exportación de información a formato PDF (extractos de cuenta y liquidaciones).
 - f. Proceso “back-end” de liquidación de tarjetas de crédito y préstamos.
2. Una aplicación bancaria enfocada al cliente:
 - a. Visualización de los productos contratados por el cliente.
 - b. Exportación de información a formato PDF (extractos de cuenta, tarjeta de crédito y préstamo).

3. Módulos formativos aplicados en el TFG

Durante el desarrollo de este TFG, se han aplicado los módulos formativos que se detallan a continuación.

3.1 Programación

Los conocimientos en esta asignatura han sido, como no podía ser de otra forma, el eje central en el desarrollo del proyecto. Lenguajes de programación, programación estructurada, programación modular, etc. son conocimientos que han sido fundamentales para poder abordar este TFG.

3.2 Desarrollo de interfaces

El desarrollo de interfaces se ha realizado con la tecnología Swing/AWT, en lenguaje Java. La aplicación desarrollada sigue el estilo de arquitectura de software “Modelo/Vista/Controlador” (MVC). Por este motivo se han separado internamente los módulos de la aplicación en paquetes, que siguen esta arquitectura de software.

3.3 Base de datos

Para el diseño de base de datos de los “**Productos Bancarios**” (“**Cuenta Corriente**”, “**Tarjeta Crédito**” y “**Préstamo**”), se ha utilizado un diseño similar al de la herencia en programación orientada a objetos. De tal forma que las tablas de los productos heredan los atributos de la tabla padre “**Producto Bancario**”.

El diseño final de la aplicación “SoftBank” dispone de un total de 13 tablas, siendo las principales “**Producto Bancario**” y “**Cliente**”.

La relación entre las tablas “**Cliente**” y “**Producto Bancario**” utiliza una relación de muchos a muchos (N:M), ya que un cliente puede tener varios productos bancarios y viceversa.

A parte de los productos mencionados, se ha contemplado el desarrollo de un cuarto producto bancario “**Avales**” que ya está incluido en el modelo de base de datos, aunque no se ha llegado a implementar.

3.4 Acceso a datos

La interacción de la aplicación con los datos almacenados en la base de datos, se ha realizado mediante el framework “Java Persistence API” (JPA), con llamadas internas agrupadas en módulos “Data Access Object” (DAO).

3.5 Entornos de desarrollo

Se ha utilizado la función de depuración “debug” para testear la aplicación y para la resolución de errores durante todo el proceso de desarrollo.

Como control de versiones se ha empleado GitHub. Aunque este proyecto, se ha desarrollado de forma individual, se ha utilizado GitHub como almacenamiento en la nube y respaldo durante el proceso de desarrollo.

4. Herramientas y Lenguajes utilizados

4.1 Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).¹

Eclipse ha sido el entorno de desarrollo más utilizado durante el ciclo formativo. Por este motivo también ha sido usado en el desarrollo de este proyecto. No obstante, por motivos que mencionaremos a continuación, su uso se ha limitado al desarrollo de módulos Java no relacionados o poco relacionados con el diseño de interfaces Swing/AWT.

El diseñador gráfico de interfaces Swing/AWT debe obtenerse a través de la descarga de un plug-in desde el "Eclipse Marketplace". Para este TFG se eligió el plug-in "WindowBuilder", que es uno de los más empleados y descargados por la comunidad. No obstante, este plug-in tiene bastantes carencias y no es muy completo, motivo por el cual ha llevado a utilizar la herramienta que se describe a continuación.

4.2 Apache NetBeans

Dadas las carencias del diseñador gráfico de Eclipse, se buscó alternativas que ofreciesen una mejor funcionalidad. Esto llevo a descargar e instalar Apache NetBeans en su versión 15.

Apache Netbeans es un entorno de desarrollo orientado a Java, desarrollado en modalidad "Open Source", de uso libre y gratuito. Este software, al igual que Eclipse tiene 4 grandes versiones al año (distribuciones trimestrales). NetBeans también tiene un sistema de plug-in's parecido al de Eclipse.

Netbeans permite el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.²

4.3 Java EE

Java es un lenguaje de programación interpretado, desarrollado inicialmente por Sun Microsystems, y actualmente es propiedad de Oracle. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de estos lenguajes. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente³.

¹ [https://es.wikipedia.org/wiki/Eclipse_\(software\)](https://es.wikipedia.org/wiki/Eclipse_(software))

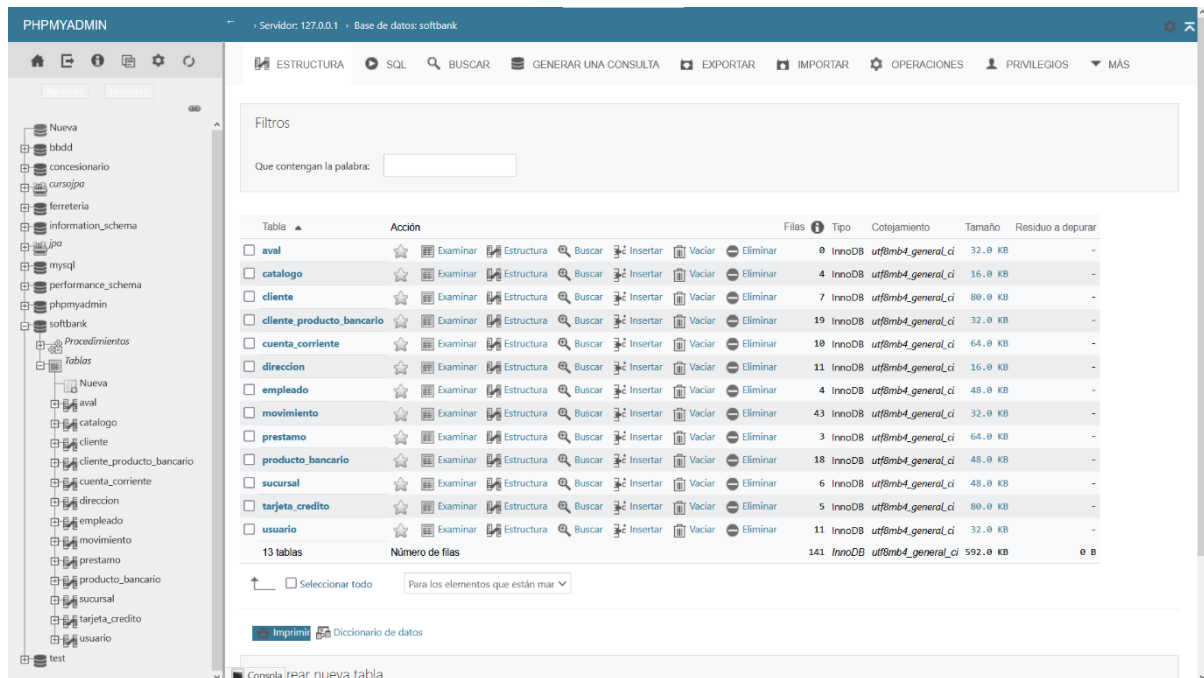
² <https://es.wikipedia.org/wiki/NetBeans>

³ [https://es.wikipedia.org/wiki/Java_\(lenguaje_de_programación\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación))

Java ha sido el principal lenguaje de alto nivel utilizado durante todo el ciclo formativo, y en el que el alumno tenía más experiencia. Por lo cual, no hubo duda en su elección para el desarrollo de este proyecto.

4.4 phpMyAdmin

Para las pruebas SQL, creación de los objetos de base de datos en MariaDB (tablas, índices, restricciones y procedimientos almacenados), y consultas interactivas, se ha utilizado la interfaz web phpMyAdmin.



1 phpMyAdmin con las tablas de "SoftBank"

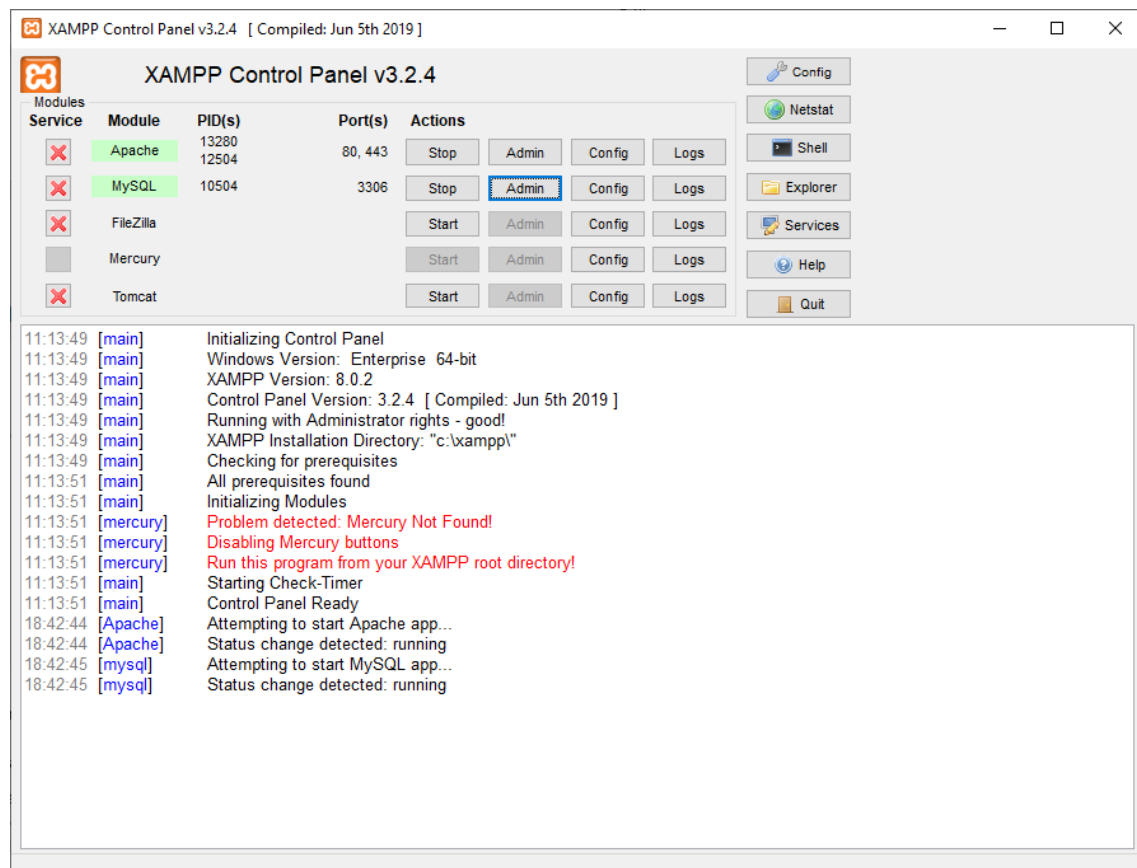
"phpMyAdmin" es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador web. Actualmente puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 72 idiomas.⁴

⁴ <https://es.wikipedia.org/wiki/PhpMyAdmin>

4.5 XAMPP

Como motor de base de datos se ha usado MariaDB, en la distribución proporcionada por XAMPP. XAMPP también proporciona el servidor de páginas web Apache, necesario para el interfaz web phpMyAdmin. XAMPP es un paquete de software libre, que se distribuye con licencia de tipo GNU (libre y gratuito).

Además de incluir el conjunto de herramientas necesario para ejecutar el motor de base de datos y el interfaz web phpMyAdmin, incluye un gestor de arranque y parada de los servicios relacionados con dichas herramientas.



2 XAMPP ejecutando Apache y MySQL

4.6 GitHub

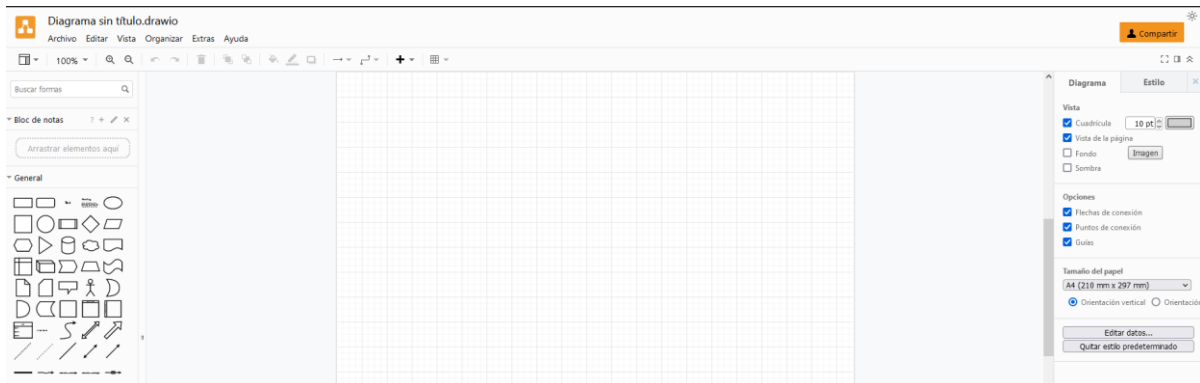
GitHub es una web para alojar proyectos de software en un repositorio en la nube. Como herramienta de control de versiones, utiliza Git. Los repositorios en la nube son lugares virtuales donde los usuarios pueden almacenar cualquier tipo de archivos. No obstante, suelen usarse para almacenar archivos que representan código de distintos lenguajes de programación.

GitHub ha sido utilizado, de forma integrada, desde los dos entornos de desarrollo (IDE) que ha usado el alumno (Eclipse y NetBeans).

Dado que los IDE son diferentes, la forma de usar GitHub tiene sus pequeñas diferencias. En Eclipse el “commit” y el “push” al repositorio son una única operación, mientras que en NetBeans deben realizarse de forma separada.

4.7 Diagrams.net

Diagrams.net⁵ es un diseñador de diagramas de uso gratuito y código abierto. Permite crear diagramas de flujo, modelado de interfaces (wireframes), organigramas y diagramas de red.



3 Pantalla inicial de Diagrams.net

La herramienta está disponible en entorno web (accesible desde el navegador) y en versión de escritorio para los sistemas operativos Linux, macOS y Windows. Esta integrada con servicios en la nube como DropBox, OneDrive y Google Drive.

Esta herramienta ha sido empleada para la creación del diagrama del modelo de datos, así como para los diagramas de clase de paneles y formularios.

4.8 JDateChooser y JCalendar

Dado que la API Swing no contaba con un componente específico para el manejo de fechas, y que el uso de “JFormattedTextField” no le pareció satisfactorio al alumno para esta tarea, se decidió usar los componentes “JDateChooser” y “JCalendar” de toedter.com⁶.

Estos componentes son de uso gratuito, y disponen de un amplio repertorio de ejemplos de manejo. Además, incluyen por defecto un botón que despliega un calendario para poder elegir una fecha de forma gráfica, lo que les hace visual y funcionalmente más interesantes.

4.9 iText 7

Es una API disponible para varios lenguajes de programación (entre ellos Java), para la creación y modificación de documentos PDF, RTF y HTML. Permite, entre otras características, rellenar formularios, mover páginas de un PDF a otro, etc.⁷

En el caso del alumno, esta API ha sido empleada para la exportación de datos a PDF de los extractos de movimientos de cuenta corriente, tarjeta de crédito y préstamos, así como las liquidaciones mensuales de las tarjetas de crédito y préstamos.

4.10 Java Persistence API (JPA)

Java Persistence API, más conocida por sus siglas JPA, es una API de persistencia desarrollada para la plataforma Java EE. El objetivo que persigue el diseño de esta API

⁵ <https://www.diagrams.net/>

⁶ <https://toedter.com/jcalendar/>

⁷ <https://itextpdf.com/products/itext-7/itext-7-core>

es no perder las ventajas de la orientación a objetos al interactuar con una base de datos⁸.

La base de uso de JPA son las entidades de persistencia (entity). Una entidad de persistencia JPA se relaciona directamente con una tabla en la base de datos. A través de anotaciones JPA (annotations) se indica la relación entre propiedades Java y objetos de la base de datos. Como ejemplos de estas anotaciones se tienen las siguientes:

- @Entity indica que la clase Java es una clase JPA. Solo podrá haber una anotación @Entity en una clase Java.
- @Table sirve para indicar a que tabla de la BB.DD. está asociada la clase Java.
- @Column indica la columna de la BB.DD. a la que hace referencia la propiedad Java.
- @Id corresponde a la propiedad Java que se relaciona con la clave primaria en la BB.DD. Solo podrá haber una anotación @Id en una clase Java.
- @GeneratedValue indica que el valor de esta propiedad Java será dado de forma automática por la BB.DD. mediante “auto increment”.
- @JoinColumn es la anotación que permite indicar las relaciones entre tablas de la base de datos.
- @NamedQuery es una anotación que permite tener de forma predefinida query’s JPA dentro de la clase Java.

Gracias a esta API, y al diseño de la base de datos realizada, se ha podido diseñar un sistema sencillo de recuperación enlazada. Por ejemplo, desde un producto bancario (cuenta corriente, tarjeta o préstamo), es muy sencillo poder recuperar toda la información relacionada (cliente al que pertenece el producto, sucursal de la cuenta, dirección del cliente, etc.).

⁸ https://es.wikipedia.org/wiki/Java_Persistence_API

5. Fases del proyecto

5.1 Idea

A la hora de decidirse por un proyecto, el alumno ha tenido en cuenta que quería desarrollar una aplicación que permitiese adquirir conceptos y conocimientos que pudiesen ser útiles en el futuro. Pensando en diferentes opciones, se le ocurrió el desarrollo de una aplicación bancaria, ya que los conocimientos adquiridos podrían servirle tanto para un uso laboral como para un uso personal. Así, conceptos como la TAE de un producto bancario (que permite la comparación entre dos productos del mismo tipo), o el IBAN de un número de cuenta (que permite la domiciliación de nómina y recibos) son útiles en ambos casos.

Además de todo lo anterior, una aplicación de este tipo le permitía escalar la complejidad de la misma, en función del tiempo disponible para el desarrollo del proyecto. También permitía abordar la extensión de forma modular, por ejemplo, podía desarrollar el producto de cuentas corrientes y después continuar con otros productos relacionados más complejos. En este caso finalmente se han abordado, cuentas corrientes, tarjetas de crédito y préstamos.

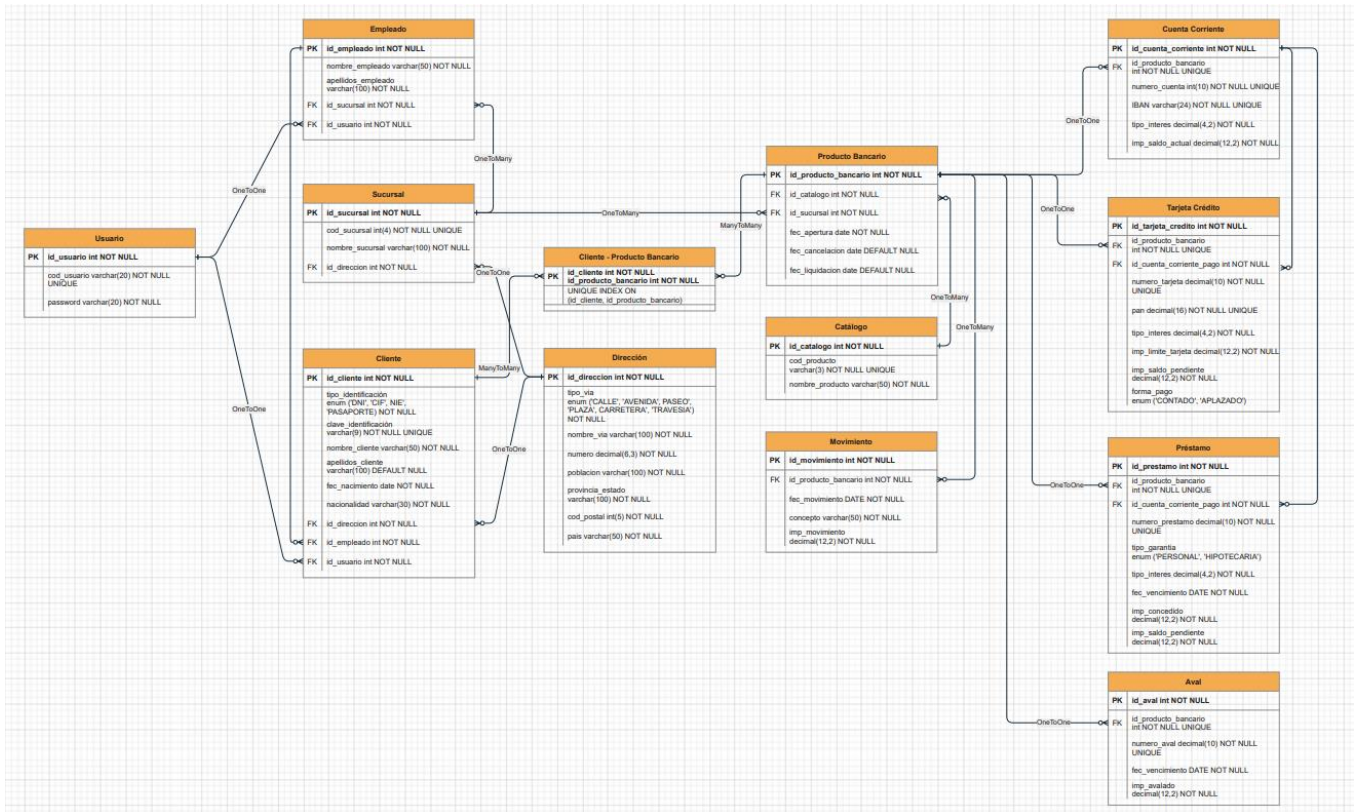
5.2 Primera aproximación

Para dejar cerrado unos primeros requisitos, el alumno se planteó que quería hacer exactamente. Lo primero que pensó, es que los bancos debían de tener unos clientes, y que estos clientes a su vez, debían estar atendidos por unos empleados. Tanto clientes como empleados, debían pertenecer a una sucursal u oficina bancaria, la cual tendría una dirección. Y por supuesto los clientes también tendrían sus propias direcciones.

A continuación, pensó en los productos que ofrecería el banco para sus clientes. Aquí entraría las cuentas corrientes, tarjetas de créditos, préstamos y avales.

5.3 Modelo de datos

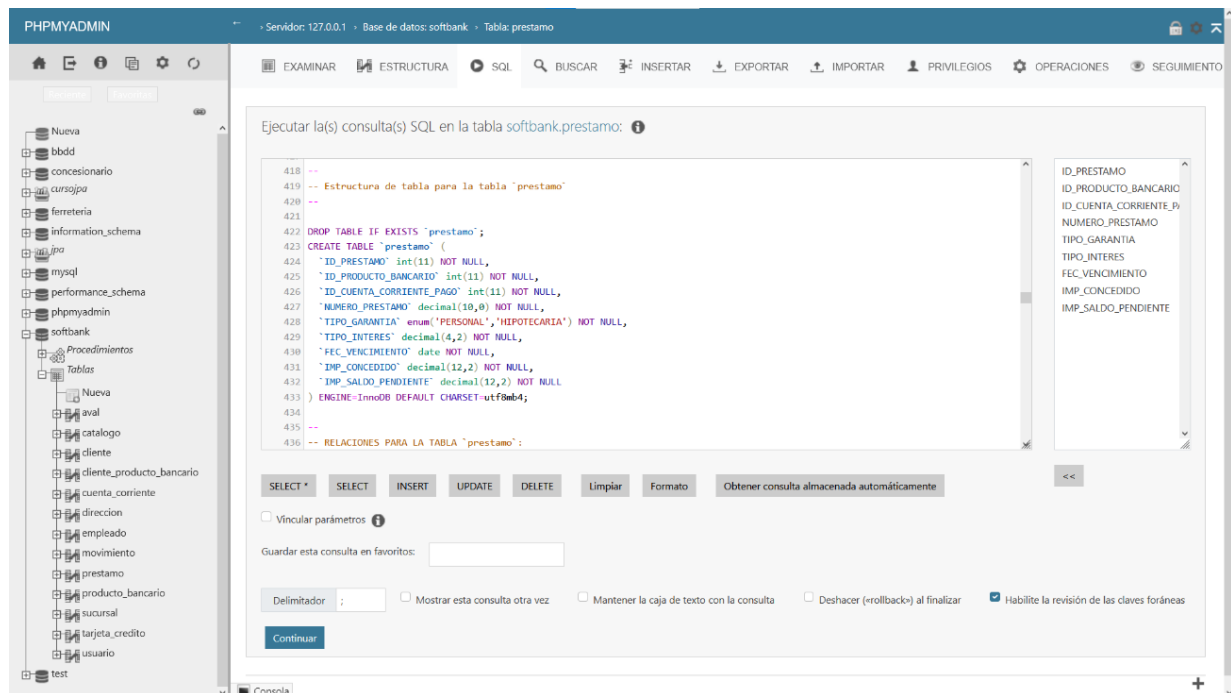
Con las premisas del punto anterior, se diseñó el siguiente modelo de datos. Este documento (“Diagrama SoftBank.pdf”) se encuentra dentro de la carpeta “doc” del proyecto, para su mejor visualización.



4 Modelo de datos de "SoftBank"

5.4 Base de datos

Para la creación de la base de datos, se ha utilizado phpMyAdmin, que es una herramienta web incorporada en el conjunto de herramientas de XAMP.



5 phpMyAdmin mostrando un fragmento SQL

El fichero SQL (“SQL SoftBank.sql”) con todas las sentencias necesarias para la creación de la base de datos, se encontrará en la carpeta “sql” dentro del proyecto.

Para la actualización de los saldos de los productos bancarios implementados, se han creado unos procedimientos almacenados en la base de datos. Cada procedimiento almacenado actualiza el saldo del producto al que corresponde.

Un procedimiento almacenado (“stored procedure” en inglés) es un programa (o procedimiento) almacenado físicamente en una base de datos. Su implementación varía de un gestor de bases de datos a otro. La ventaja de un procedimiento almacenado es que es ejecutado directamente en el motor de bases de datos, que habitualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

Al incluir la lógica de la aplicación en la base de datos utilizando procedimientos almacenados, la necesidad de incluir la misma lógica en todos los programas que acceden a los datos se reduce. Esto puede simplificar la creación y, particularmente, el mantenimiento de los programas involucrados.⁹

Se han utilizado claves foráneas para asegurar la integridad de los datos almacenados, así como índices que evitan la duplicidad de datos clave.

5.5 Modelo, Vista y Controlador (MVC)

Se ha decidido usar el patrón de arquitectura software modelo-vista-controlador. Este patrón separa la lógica de la aplicación, permitiendo la reutilización de código y facilitando su implementación y mantenimiento.

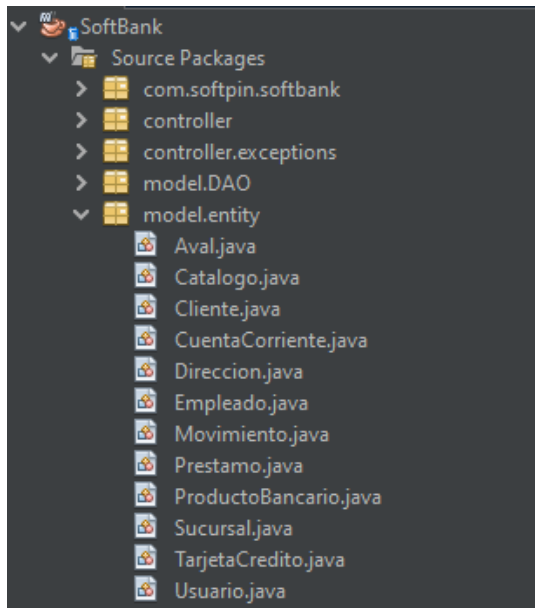
La aplicación se creó, dentro del IDE Eclipse, como un proyecto “Maven”. Esto ha facilitado enormemente el uso de las distintas librerías utilizadas en la aplicación (iText, mysql, jcalendar, etc).

El proyecto utiliza JPA como framework de acceso a datos. Para la creación de las clases relacionadas con las tablas de la base de datos (“entity classes”), se ha utilizado el IDE NetBeans. Este IDE ha creado de forma semi automática, los módulos JPA Controller, que contienen la lógica para mantener correctamente relacionadas las entidades de la base de datos. Estos JPA Controller han tenido que ser, en ocasiones, ligeramente retocados para garantizar su correcto funcionamiento.

⁹ https://es.wikipedia.org/wiki/Procedimiento_almacenado

5.5.1 Modelo

5.5.1.1 Modelo Entity



6 Clases "JPA Entities"

Este paquete Java contiene las clases que se relacionan directamente con sus tablas en la base de datos. Estas clases han sido en su mayor parte autogeneradas por el IDE NetBeans.

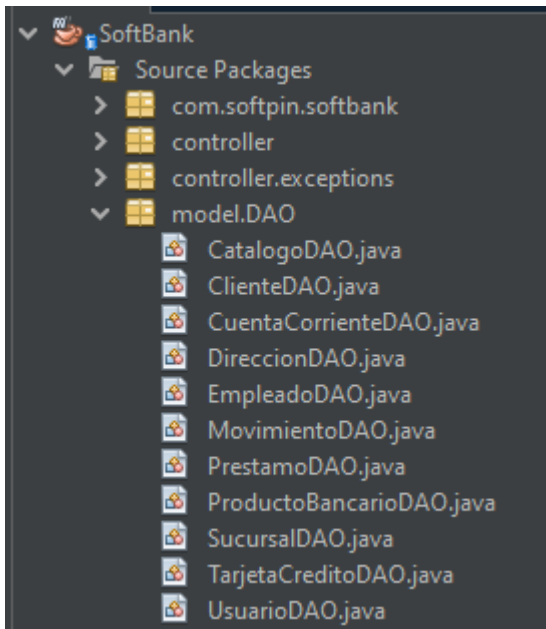
Estas clases contienen todas las etiquetas JPA pertinentes para la identificación de la clave primaria, claves foráneas, nombres de columnas, nombre de tabla, etc. También contienen los “setters” y “getters” para poder acceder a las clases relacionadas.

Se ha tenido que modificar en algunos casos, el método “equals()” para que no solo compare la propiedad “id” del objeto, si no también el resto de propiedades. También se ha modificado el método “toString()”, para

que se muestren todas las propiedades de la clase.

Todas las relaciones entre entidades son bidireccionales, habiéndose comprobado que se han declarado según lo esperado en la base de datos. Es decir, las relaciones 1:N se han declarado como @OneToMany, las relaciones M:N se han declarado como @ManyToMany, y las relaciones 1:1 como @OneToOne.

5.5.1.2 Modelo DAO



7 Clases "Data Access Object" (DAO)

Para aislar la lógica de acceso a datos del resto de la aplicación se ha implementado el patrón de arquitectura software “Data Access Object” (DAO).

Esta arquitectura permite ser más flexible a la hora de elegir el motor de base de datos donde se almacena la información. Así, en el futuro sería muy sencillo cambiar de origen de información (por ejemplo, cambiar de motor de base de datos).

Los DAO proporcionan los métodos para la inserción, actualización, eliminación y consulta de la información almacenada en la base de datos.

```

/**
 * Método para el alta de un nuevo {@link Cliente}.
 *
 * @param c El objeto {@link Cliente} a crear.
 * @return <b>true</b> El {@link Cliente} ha sido dada de alta.
 * <br><b>false</b> El {@link Cliente} no ha sido dada de alta.
 */
public boolean insertarCliente(Cliente c) {
    estado = false;
    mensaje = "insertarCliente -> ";
    try {
        c.setIdCliente(idCliente:null);
        cjc.create(c);
        estado = true;
        mensaje = mensaje + "OK: " + c;
        cliente = c;
    } catch (Exception E) {
        mensaje = mensaje + "ERROR: " + c + "\n" + E.getMessage();
        System.out.println(":" + mensaje);
    }
    return estado;
}

```

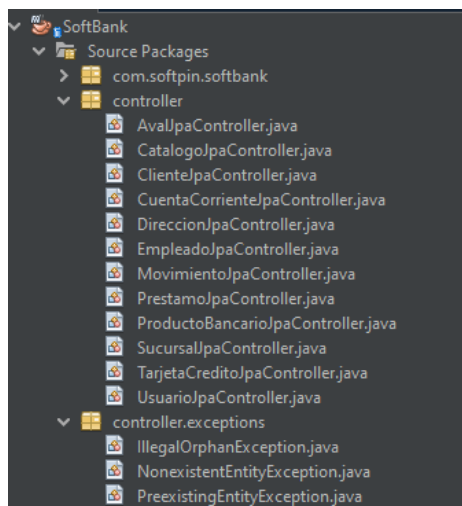
8 Fragmento de ClienteDAO.java

5.5.2 Vista

La Vista tiene la función de presentar datos al usuario y recoger la información que este proporciona. En el caso de este proyecto, la Vista corresponde con la interfaz gráfica de usuario. La Vista interactúa con el Modelo a través de los DAO y de los JPA Controller.

En esta aplicación la vista se ha implementado mediante la API Swing. En un punto posterior se explicará en detalle.

5.5.3 Controlador



Esta capa de la arquitectura MVC es la encargada de mantener la integridad entre las entidades del proyecto. De esta forma cuando se inserta, actualiza o elimina un elemento en la base de datos, el controlador se encarga de actualizar las referencias relacionadas con los otros objetos JPA.

Como ejemplo se muestra el código correspondiente a la inserción de un usuario:

```
public void create(Usuario usuario) {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Cliente cliente = usuario.getCliente();
        if (cliente != null) {
            cliente = em.getReference(entityClass: cliente.getClass(), primaryKey: cliente.getIdCliente());
            usuario.setCliente(cliente);
        }
        Empleado empleado = usuario.getEmpleado();
        if (empleado != null) {
            empleado = em.getReference(entityClass: empleado.getClass(), primaryKey: empleado.getIdEmpleado());
            usuario.setEmpleado(empleado);
        }
        em.persist(entity: usuario);
        if (cliente != null) {
            Usuario oldUsuarioOfCliente = cliente.getUsuario();
            if (oldUsuarioOfCliente != null) {
                oldUsuarioOfCliente.setCliente(cliente: null);
                oldUsuarioOfCliente = em.merge(entity: oldUsuarioOfCliente);
            }
            cliente.setUsuario(usuario);
            cliente = em.merge(entity: cliente);
        }
        if (empleado != null) {
            Usuario oldUsuarioOfEmpleado = empleado.getUsuario();
            if (oldUsuarioOfEmpleado != null) {
                oldUsuarioOfEmpleado.setEmpleado(empleado: null);
                oldUsuarioOfEmpleado = em.merge(entity: oldUsuarioOfEmpleado);
            }
            empleado.setUsuario(usuario);
            empleado = em.merge(entity: empleado);
        }
        em.getTransaction().commit();
    } finally {
        if (em != null) {
            em.close();
        }
    }
}
```

10 Método "create" en UsuarioJPAController.java

Estos módulos han sido autogenerados por el IDE NetBeans, aunque han sido modificados en algunos casos.

5.6 Java

Java ha sido el lenguaje más usado durante todo el ciclo formativo, por tanto, era la opción más obvia para abordar el desarrollo de este proyecto. Además, Java es uno de los lenguajes más empleados en el ámbito laboral, por lo que toda experiencia que se pudiera adquirir en TFG sería útil en el futuro.

Para el desarrollo de este proyecto se ha empleado la versión 17.0.5 del Software Development Kit (SDK) Java.

5.7 Swing

Una vez elegido Java como lenguaje de programación para este proyecto, también parecía obvia la API (Application Program Interface) para el desarrollo y diseño de interfaces. Swing es la API por defecto que proporciona la plataforma Java y aunque esta siendo progresivamente reemplazada por JavaFX (la API gráfica más reciente), sigue siendo la más usada por la comunidad de desarrolladores.

Además, Swing es la API de la que se pueden encontrar mayor número de ejemplos y la que tiene mayor número de dudas resueltas en páginas web de consulta como <https://stackoverflow.com/>.

Swing proporciona objetos gráficos, tales como botones, paneles, listas desplegables, tablas, etc; que permiten construir una interfaz gráfica de ventanas para el usuario. Swing es una API que evolucionó desde AWT, el API gráfico original de la plataforma Java. Una de sus ventajas reside en el hecho de que Swing es independiente de la plataforma en la que se ejecuta, por lo que el diseño de las ventanas funcionará exactamente igual en un entorno Windows, que en un entorno Unix/Linux.

5.8 Diseño de Interfaces

La premisa principal a la hora del diseño de las interfaces es que estas fueran lo más reutilizables posibles. Es decir, que fueran programadas una vez y usadas todas las veces que fueran necesarias. El alumno decidió que la mejor forma de lograr este objetivo era ejecutar un diseño basado en paneles Swing.

Estos paneles contendrían la lógica de presentación y recogida de información relacionada con una entidad JPA (que será la encargada de la persistencia en la BB.DD.) Así, por ejemplo, para la entidad “Cliente” existiría un panel dedicado exclusivamente a esta entidad.

También el alumno pensó que, para facilitar el desarrollo y reutilización de estos paneles, debían de contar con un diseño muy similar y con una serie de métodos y propiedades que fuesen comunes. Por este motivo, los paneles debían heredar de una clase diseñada por el alumno que contuviese esos métodos y propiedades comunes.

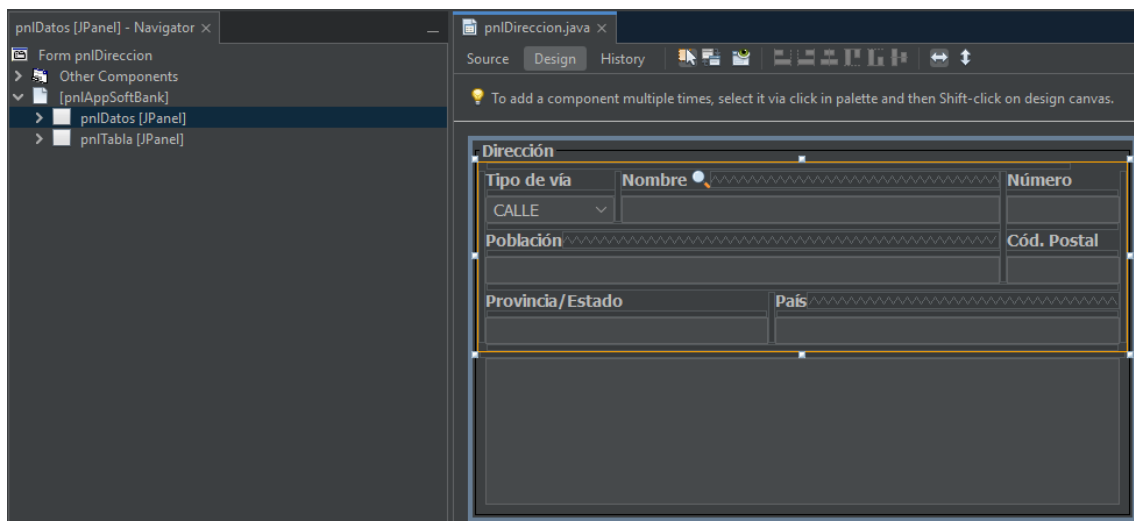
Con estas premisas en mente, el alumno diseñó una estructura de panel que consta de dos partes: una en la que se mostraría y se recogería la información detallada de la clase, y otra (con estructura de tabla) que mostrase la información fundamental y que permitiese la selección por parte del usuario. Estas partes serían subpaneles que se llamarían, respectivamente, “**pnlDatos**” y “**pnlTabla**”.

A su vez, el alumno tenía claro que, aunque los paneles contendrían la lógica correspondiente a su entidad JPA, no podrían funcionar sino es bajo el contenedor de un formulario (“JFrame”). Este formulario debería poder trabajar de forma conjunta con los paneles diseñados y ofrecer una serie de acciones (“Insertar”, “Modificar”, “Eliminar”) que fuesen comunes a los paneles que contenga. Con todos estos condicionantes, el alumno ha realizado el diseño que se explica a continuación.

Para los paneles se ha diseñado una clase “padre”, llamada “**view.pnlAppSoftBank**”, que implementa las siguientes propiedades comunes a los paneles de la aplicación:

- **emf**: propiedad que recoge la conexión con la BB.DD. Por tanto, todos los paneles compartirán la misma conexión.
- **appUser**: esta propiedad contiene el objeto de clase usuario conectado a la aplicación. Determinará, entre otros, que elementos puedan ser visualizados y/o modificados. Así, por ejemplo, un “cliente” sólo podría ver la información propia, mientras que un “empleado” tendría capacidad de visualizar la información de todos los “clientes”.

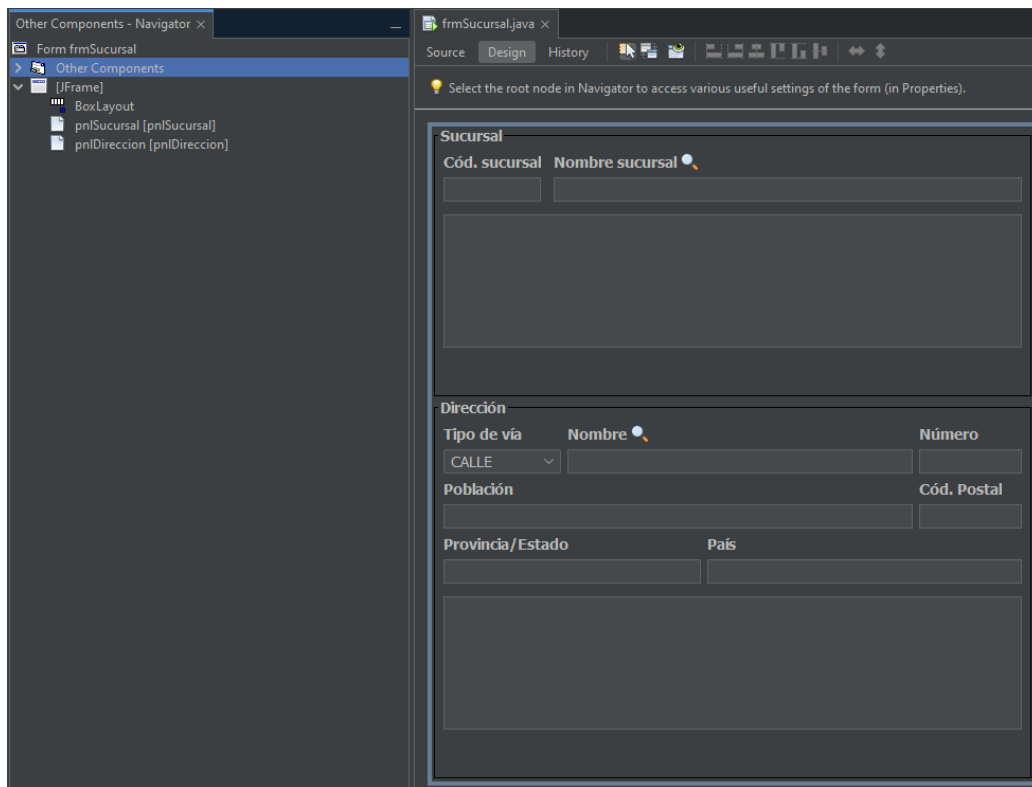
- **jpaObject**: propiedad que recoge el objeto JPA que está manejando el panel.
- **panelMode**: propiedad que determina que elementos del panel serán accesibles. Esta propiedad puede tener tres valores:
 - o **CRUD**: indica que el panel está en modo de alta, baja y modificación. Por tanto, todos los elementos serán accesibles y modificables.
 - o **SELECTION**: indica que el panel está en modo selección y los elementos serán accesibles, pero no modificables.
 - o **READ_ONLY**: indica que el panel está en modo de lectura y los elementos sólo serán visualizables. Además, por defecto, ocultará el subpanel “**pnlTabla**”.
- **frmParent**: propiedad que recoge el formulario (“JFrame”) en el que está embebido el panel. Esta propiedad permitirá el intercambio de información entre los paneles y su contenedor, así como la presentación de información en una barra de estado (“Status Bar”).



11 Panel de Dirección (pnlDireccion) y sus subpaneles

Para los formularios se ha diseñado una clase “padre”, llamada “**view.frmAppSoftBank**”, que implementa las siguientes propiedades comunes a los formularios de la aplicación:

- **emf, appUser y jpaObject**: son las mismas propiedades de los paneles antes descrita. El formulario las recibe y las transmite a todos los paneles que contiene.
- **formMode**: es la misma propiedad de los paneles antes descrita. El formulario la recibe y la transmite, según corresponda, a los paneles que contiene.
- **formAction**: propiedad específica para los formularios de productos bancarios y que determina la acción que se está realizando. Esta propiedad tiene cuatro valores posibles:
 - o **NEW**: indica que se está creando un nuevo producto.
 - o **CANCEL**: indica que se está cancelando un producto ya existente.
 - o **QUERY**: indica que se está consultado información de un producto ya existente.
 - o **MOVEMENT**: indica que se está realizando altas, bajas y modificaciones de movimientos de un producto bancario ya existente.

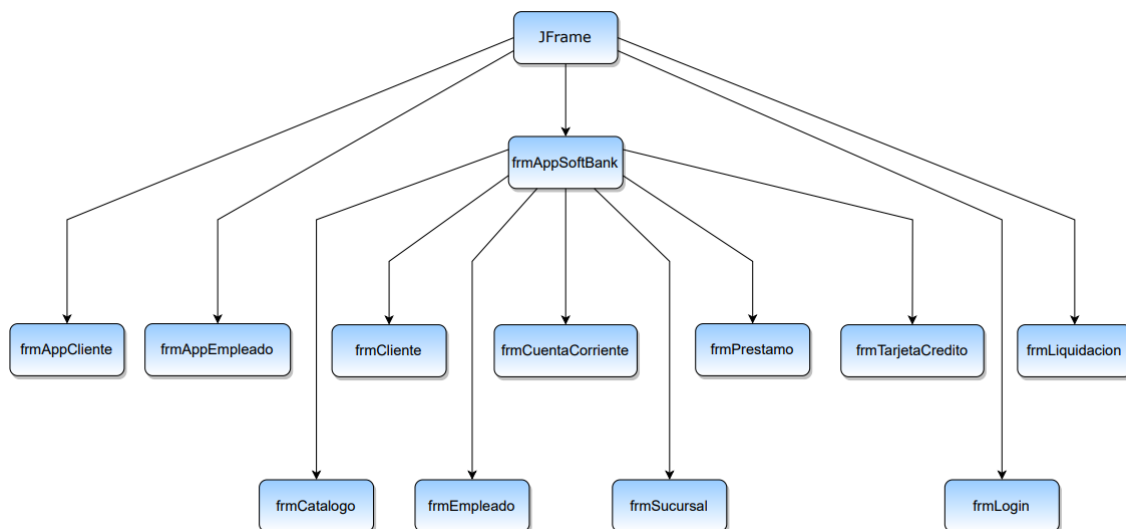


12 Formulario de Sucursales (frmSucursal) y sus paneles

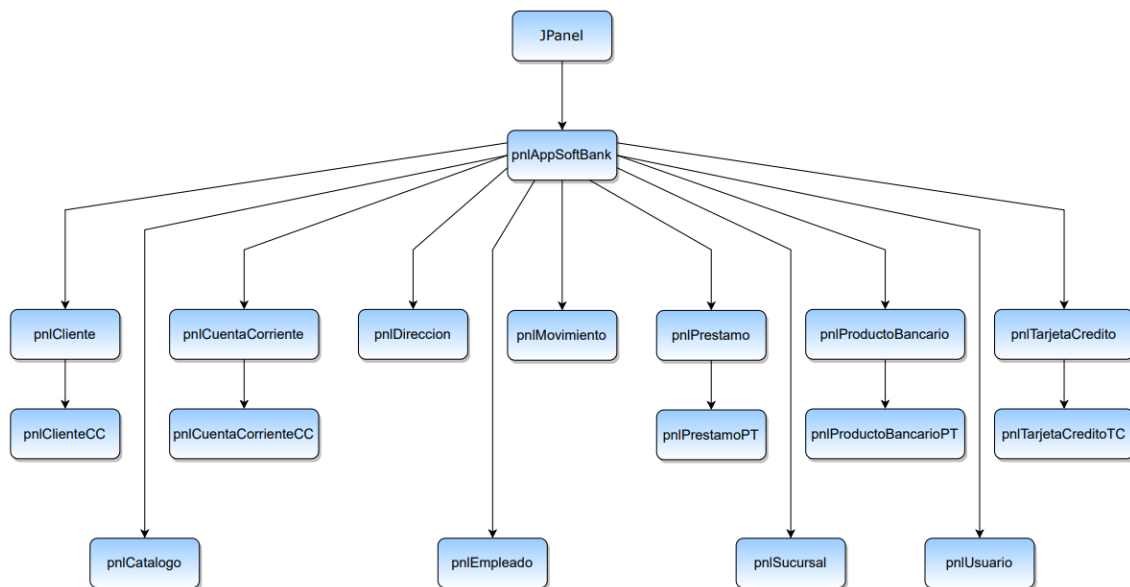
Tras completar el diseño de los paneles y formularios, se han obtenido un total de 17 paneles y 12 formularios.

5.9 Diagramas de clases

Tras la finalización el diseño, se pudo obtener un diagrama de las clases utilizadas. A continuación, se muestra el desglose de las clases de tipo formulario ("JFrame") y de las de tipo panel ("JPanel"). Estos documentos ("Diagrama JFrame.pdf" y "Diagrama JPanel.pdf") se encuentran dentro de la carpeta "doc" del proyecto, para su mejor visualización.



13 Diagrama de clases "JFrame"



14 Diagrama de clases "JPanel"

5.10 Validaciones relacionadas con el entorno bancario

Además de las complejidades informáticas del proyecto, el alumno sabía desde el principio que habría que obtener una serie de conocimientos de negocio bancario para poder abordar la aplicación. Por este motivo, se han incorporado a la aplicación los elementos de negocio que se detallan a continuación.

5.10.1 CCC (Código de Cuenta de Cliente)

El CCC es un número de 20 dígitos que identifica el número de cuenta en el sistema bancario español. Este código apareció en 1979 y fue impulsado por el Consejo Superior Bancario (CSB) para poder identificar, de forma única, cualquier cuenta del sistema bancario español.

Su estructura es la siguiente:

Código de entidad bancaria	Código de sucursal bancaria	Dígitos de control	N.º de cuenta
4 dígitos	4 dígitos	2 dígitos	10 dígitos

El código de entidad bancaria es único para cada banco y es asignado por el Banco de España. Ejemplos de este número sería el 0182, que identifica a BBVA. O el 0049 que identifica a Banco de Santander. Para este proyecto, se ha designado el número 0138 que coincide con el de Bankoia en España. Se ha elegido un número de banco válido para poder comprobar, con herramientas externas ¹⁰, que el número de CCC generado es sintácticamente válido.

El código de sucursal es propio de cada entidad bancaria y en el TFG se asigna de forma libre.

¹⁰ <http://www.aplicacionesinformaticas.com/programas/gratis/ctabanco.php>

A continuación, están los 2 dígitos de control que se utilizan de la siguiente manera:

- El primer dígito de control sirve para controlar el código de entidad bancaria y el código de sucursal. Es un “checksum” (suma de verificación) especial de estos primeros 8 dígitos.
- El segundo dígito de control sirve para contralar el número de cuenta. Es, por tanto, un “checksum” especial de los últimos 10 dígitos del CCC.

Por último, el número de cuenta también es de libre asignación por cada entidad bancaria y, en el caso aplicado en este proyecto empieza desde el número 1000000001.

En la aplicación, el CCC se calcula desde el método “**getCCC()**” situado en la clase “**utilities.Banking**”.

5.10.2 IBAN (International Bank Account Number)

El IBAN es una secuencia alfanumérica de 24 caracteres, que identifica un número de cuenta en la zona SEPA (Single Euro Payment Area). IBAN es el acrónimo de International Bank Account Number (número de cuenta bancaria internacional) y es un código bancario que sirve para identificar cada cuenta corriente de manera única en la zona SEPA que, según el comité europeo, comprende 34 países (entre ellos los 28 de la Unión Europea).

Gracias al código IBAN, una persona puede realizar pagos o transferencias en cualquiera de los países de la zona SEPA de igual forma que si estuvieran haciendo un pago nacional.

La estructura de un código IBAN de una entidad bancaria española es la siguiente:

Código de país	Dígitos de control	CCC
2 caracteres. En el caso de España, son “ES”.	2 dígitos	20 dígitos.

Los dígitos de control se calculan en base a todo el contenido del IBAN, incluyendo a los caracteres que identifican al país. En la aplicación, el IBAN se calcula desde el método “**getIBAN()**” situado en la clase “**utilities.Banking**”.

Comentar que los IBAN generados en la aplicación son válidos sintácticamente, y pueden ser comprobados mediante herramientas externas¹¹.

5.10.3 PAN (Personal Account Number)

El PAN es una secuencia numérica de 16 dígitos que identifica de forma única a una tarjeta de crédito (o débito). Este número identifica tanto al sistema emisor de la tarjeta (Visa, MasterCard, American Express, etc) como a la entidad emisora de la tarjeta (banco u entidad financiera).

La estructura de un número PAN es la siguiente:

Sistema emisor	Entidad emisora	N.º de tarjeta	Dígito de control
1 dígito	4 dígitos	10 dígitos	1 dígito

¹¹ <https://es.iban.com/>

En este proyecto se ha elegido como sistema emisor a “MasterCard”, por lo que todas las tarjetas de SoftBank empezarán por el número “5”. La entidad emisora corresponde a la entidad de SoftBank (0138).

Los 10 dígitos de número de tarjeta son de libre disposición para cada banco. En esta aplicación el alumno eligió que empezarán desde el número 4000000001.

Por último, el dígito de control se calcula a partir de los 15 dígitos previos. Se utiliza un algoritmo usado en diferentes tipos de validación, conocido como “Algoritmo de Luhn”¹². Brevemente explicado, es un algoritmo que suma de forma independiente los números pares e impares que componen el número a validar. Después, a la suma obtenida, se le aplica un módulo 9.

En la aplicación, el PAN se calcula desde el método “**getPAN()**” situado en la clase “**utilities.Banking**”. Los PAN generados por la aplicación son sintácticamente correctos y pueden comprobarse mediante herramientas externas¹³.

5.10.4 TAE (Tasa Anual Equivalente)

La TAE es la cifra que permite la comparación directa de la rentabilidad entre productos financieros del mismo tipo, pero de diferentes entidades bancarias. La TAE se calcula en base al tipo de interés ofrecido por el producto bancario (cuenta corriente, depósito, préstamo, tarjeta de crédito, etc.) y debe incluir los gastos y comisiones inherentes al producto. Así, por ejemplo, comparando la TAE de dos depósitos de distintas entidades bancarias se sabe que se ganaría más dinero con el que tenga la TAE más alta. En el caso de los préstamos, se pagará menos dinero con el préstamo que ofrezca la TAE más baja. En este proyecto, la TAE calculada solo tiene como base el tipo de interés del producto bancario, ya que no se han incluido gastos de ningún tipo.

Para los préstamos, se ha utilizado un sistema de amortización llamado “amortización francesa”. Este tipo de amortización se caracteriza por tener una cuota constante y un tipo de interés compuesto (ya que se hacen pagos mensuales que disminuyen el capital a pagar). Por tanto, para el cálculo de la TAE en los préstamos se realiza un cálculo compuesto que se obtiene desde el método “**getCompoundTAE()**” situado en la clase “**utilities.Banking**”.

Para las tarjetas de crédito, se utiliza un sistema de amortización directa. Por tanto, su TAE tiene un cálculo diferente y se realiza en el método “**getTAE()**” situado en la clase “**utilities.Banking**”.

Comentar que tanto la TAE¹⁴ de los préstamos como el cuadro de amortización¹⁵ que se puede obtener en la aplicación han sido validados con los cálculos que ofrece el portal del cliente de Banco de España.

¹² https://es.wikipedia.org/wiki/Algoritmo_de_Luhn

¹³ <https://www.validcreditcardnumber.com/>

¹⁴ https://app.bde.es/asb_www/es/tae.html#/principalTAE

¹⁵ https://app.bde.es/asb_www/es/cuota.html#/principalCuota

5.11 Validación de documentos de identificación

Para la creación de los nuevos clientes en la aplicación, se ha incluido como dato clave su documento de identificación. Este documento, que puede ser de varios tipos, exige también una validación para evitar errores en su introducción.

A continuación, se expone brevemente los tipos de documentos válidos y las validaciones aplicadas a los mismos.

5.11.1 DNI

El DNI (Documento Nacional de identidad), que identifica a las personas físicas españolas, es una secuencia compuesta de 8 dígitos y una letra de control. Esta letra de control se obtiene en base al módulo 23 de los 8 dígitos del DNI. En función del módulo obtenido, existe una tabla de conversión¹⁶ a la letra que figura en el documento.

A título informativo, la tabla de conversión es la siguiente:

RESTO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LETRA	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

En la aplicación, el DNI se valida con el método “**validarDNI()**” situado en la clase “**utilities.CheckIdDocuments**”.

5.11.2 CIF

El CIF (Código de Identificación Fiscal), que identifica a las personas jurídicas españolas, es una secuencia compuesta de 1 letra, seguida de 7 dígitos, y un número o letra de control. El número o letra de control se asigna en función de los 8 caracteres anteriores. Será un dígito de control cuando el CIF comienza por cualquiera las letras A, B, C, D, E, F, G, H, I, o J. Será una letra de control cuando el CIF comienza por cualquiera de las letras K, P, Q, R, S, N, o W¹⁷.

En la aplicación, el CIF se valida con el método “**validarCIF()**” situado en la clase “**utilities.CheckIdDocuments**”.

5.11.3 NIE

El NIE (Número de Identificación de Extranjero), que identifica a las personas físicas extranjeras residentes en España, es una secuencia compuesta de 1 letra, 7 dígitos y una letra de control. La primera letra sólo podrá ser una X, Y o Z. La letra de control se obtiene en base al módulo 23 de los 8 caracteres previos del NIE. La tabla de conversión para la letra de control es la misma que se utiliza para el DNI.

En la aplicación, el NIE se valida con el método “**validarNIE()**” situado en la clase “**utilities.CheckIdDocuments**”.

5.12 Exportación a PDF

Durante el desarrollo del proyecto se hizo evidente la necesidad de exportar información a un formato que luego pudiera imprimirse. Los extractos de cuenta corriente, tarjeta de crédito y préstamo, así como los cuadros de amortización de los préstamos exigían esta

¹⁶ <https://www.letranif.com/formula-para-calcular-nif/>

¹⁷ <https://www.clasesdeinformaticaweb.com/programas-en-java/cif-calculador-el-cif-en-java/>

funcionalidad. Por tanto, se decidió que lo más conveniente era exportar esta información a formato PDF, de forma que pudiera visualizarse e imprimirse una vez generada.

Para conseguir este objetivo, se plantearon dos posibilidades. Una era utilizar la API Apache PDFBox. Como ventajas de esta API son que es Open-source y de uso completamente gratuito. Sin embargo, no tiene una base de uso muy amplia, los ejemplos de uso que pueden encontrarse no son demasiados y no tiene implementada la inclusión de “tablas” dentro del documento PDF.

La otra opción era utilizar la API iText 7, cuyas ventajas son que tiene una base de uso muy amplia, y que dispone de multitud de ejemplos de uso. Además, dispone de forma nativa el uso de “tablas” dentro del documento PDF. Como inconveniente, se trata de una API comercial, aunque dispone de una licencia de uso no comercial, que es la empleada en el desarrollo del proyecto.

En la aplicación se han implementado dos módulos de generación de documentos PDF, ambos ubicados en el paquete “pdf” del proyecto. Son los siguientes:

- ItxAmortization.java
 - o Este módulo crea un documento PDF con el cuadro de amortización francesa calculado para un préstamo. Recibe como parámetro una instancia de la clase **ItxAmortizationParams** que contiene los datos necesarios para este cálculo. El resultado se genera en la carpeta “results/prod” del proyecto.

Pago	Cuota	Capital	Intereses	Pendiente
1	2.012,51 €	1.993,76 €	18,75 €	4.006,24 €
2	2.012,51 €	1.999,99 €	12,52 €	2.006,24 €
3	2.012,51 €	2.006,24 €	6,27 €	0,00 €
TOTAL	6.037,54 €	6.000,00 €	37,54 €	

15 Cuadro de amortización de préstamo

- ItxDocument.java
 - o Este módulo crea dos tipos diferentes de documentos PDF. En primer lugar, es capaz de crear los extractos de cuenta corriente, tarjeta de crédito y préstamo. En estos extractos se muestran el detalle de los movimientos de estos productos.

SOFTBANK

Sucursal (1000) MADRID - OFICINA PRINCIPAL
 CALLE ALCALÁ, 75
 28009 MADRID
 MADRID - ESPAÑA


 Extracto de Tarjeta de Crédito
 

D/DÑA. FRANCISCO JIMÉNEZ DÍAZ
 AVENIDA PIO XII, 2
 46009 VALENCIA
 VALENCIA - ESPAÑA
 PAN: 5013 8400 0000 0011 / Tarjeta: 40000000001 / Pago: CONTADO / Límite: -2.000,00 €
 Su gestor: JOSÉ SÁNCHEZ PÉREZ, en sucursal (1001) MADRID - A.G. 1 (VALLECAS)


Fecha	Concepto	Imp. Deudor	Imp. Acreditor
01/10/2022	SALDO INICIAL		0,00 €
15/10/2022	COMPRA CARREFOUR VILLAVERDE	-123,25 €	
16/10/2022	COMPRA AMAZON PED. 1134234	-70,50 €	
02/11/2022	COMPRA DECATHLON	-75,25 €	
08/11/2022	ITV CTM VALLECAS	-75,00 €	
08/12/2022	SALDO FINAL	-344,00 €	

*** Condiciones legales de Tarjeta de Crédito ***



Tipo de interés aplicado 20%. TAN 30,38%. La forma de pago APLAZADO implica el cobro del 20% del saldo pendiente, con un valor mínimo de 20€.

La forma de pago CONTADO implica el cobro completo a la fecha de liquidación. En caso de no haber cobro suficiente en la cuenta de cobro designada, el saldo pendiente quedará acumulado en la tarjeta. Pueden aplicarse comisiones de cambio y de reclamación de saldos.

- También genera los documentos de liquidación mensual de tarjetas de crédito y de préstamo. En estos documentos se muestra la liquidación de intereses y capital de estos productos.



Sucursal (4020) CÁDIZ - CHICLANA DE LA FRONTERA
 CARRETERA DE LA BARROSA, 35.5
 11139 CHICLANA DE LA FRONTERA
 CÁDIZ - ESPAÑA


Liquidación de Préstamo a fecha 30/06/2022


D./DÑA. CHICLANA NATURAL, S.A.
 CALLE DOCTOR PEDRO VELEZ, 5
 11130 CHICLANA DE LA FRONTERA
 CÁDIZ - ESPAÑA

Préstamo: 30500000003 / Garantía: PERSONAL / Imp.: -8.000,00 € / Vto.: 03/09/2022

Su gestor: ADRIÁN VALLE HERNÁNDEZ, en sucursal (4020) CÁDIZ - CHICLANA DE LA FRONTERA

Fecha	Concepto	Imp. Deudor	Imp. Acreedor
03/06/2022	SALDO INICIAL		0,00 €
03/06/2022	PRÉSTAMO GAR. PERSONAL CONCEDIDO	-8.000,00 €	
30/06/2022	INTERESES PRÉSTAMO	-18,87 €	
30/06/2022	AMORTIZACIÓN PRÉSTAMO		1.811,38 €
30/06/2022	SALDO FINAL	-4.205,81 €	

Cuenta de cargo: ES56 0138 4020 9710 0000 0010

A título informativo, la amortización de capital de este mes ha sido de 1.794,39 €.

*** Condiciones legales de Préstamo ***

Tipo de interés aplicado 3,75%. TAE 3,83%. El prestatario respalda el pago del préstamo (incluyendo intereses, gastos y comisiones) con todos sus bienes personales, tanto presentes como futuros. En la garantía HIPOTECARIA, además, se incluye expresamente el inmueble objeto del préstamo. Prestatista y prestario quedan sometidos a la jurisdicción de los juzgados de Madrid.

17 Extracto de liquidación de préstamo

6. Conclusiones y mejoras del proyecto

El alumno ha adquirido una serie de conocimientos, técnicos y funcionales, que serán de utilidad tanto en el ámbito personal como en el laboral. Más concretamente, en el desarrollo de interfaces gráficas Swing, el alumno ha ganado una gran experiencia que podrá ser explotada en el futuro.

Como demostrador tecnológico, la aplicación ha incluido una serie de capacidades que el alumno no había utilizado hasta ahora y que le han permitido agregar conocimientos no impartidos en el ciclo formativo, como, por ejemplo, toda la información relacionada con el ámbito bancario.

Después de haber terminado la aplicación, el alumno habría mejorado los siguientes aspectos:

- Se habría incluido el producto bancario “Avaless”, que estaba contemplado en el modelo de datos pero que, por falta de tiempo, no ha podido implementarse.
- Se hubiera mejorado la interfaz gráfica, para hacerla más agradable visualmente. Posiblemente, se hubiera utilizado JavaFX en lugar de Swing como API gráfica.
- Se habría incluido la posibilidad de que los clientes puedan cambiar su contraseña de acceso a la aplicación.
- Se habría incluido la exportación de información a otros formatos electrónicos, como Excel.
- Se habría incluido también la posibilidad de enviar la información por correo electrónico a los clientes.

7. Bibliografía

A lo largo del proyecto se han incluido notas al pie de pagina indicando las referencias externas utilizadas. No obstante, de forma genérica, se han utilizado las fuentes que se detallan a continuación.

- Wikipedia (<https://es.wikipedia.org>)
- StackOverflow (<https://stackoverflow.com/>)
- JCalendar y JDateChooser (<https://toedter.com/>)
- FlatIcon (<https://www.flaticon.com/>)
- iText Jump Start Tutorial (<https://kb.itextsupport.com/home/it7kb/ebooks/itext-7-jump-start-tutorial-for-java>)
- Portal del Cliente Bancario del Banco de España (<https://clientebancario.bde.es/pcb/es/>)
- Edix (<https://campus.edix.com>)

8. Anexos

En la misma carpeta donde se encuentra este documento (carpeta “doc” del proyecto), se incluyen también los siguientes anexos:

- Guía del usuario (“Guía de Usuario SoftBank.pdf”).
- Diagrama de modelo de datos (“Diagrama SoftBank.pdf”).
- Diagrama de clases JPanel (“Diagrama JPanel.pdf”).
- Diagrama de clases JFrame (“Diagrama JFrame.pdf”).

Otros ficheros útiles para la configuración y ejecución de la aplicación son:

- Sentencias SQL para la creación de la BB.DD. “SoftBank” (fichero “SQL SoftBank.sql”, situado en la carpeta “sql” del proyecto).
- Fichero de lotes Windows para la ejecución de la aplicación (fichero “SoftBank.bat”, situado en la carpeta “bat” del proyecto).