Práctica de Diseño de Software

Grado en Ingeniería Informática
Mención Ingeniería del Software
Universidad de Valladolid

curso 2019-2020

Capítulo 1

Supuesto práctico

Para consultar las normas de entrega vaya al final del documento (página 20).

1.1. Punto de partida para requisitos y análisis

En este proyecto se persigue realizar el diseño e implementación (parcial) correspondiente a la informatización de una vinoteca. La empresa actualmente se dedica a la venta de vino a sus abonados directamente en la tienda o a través del teléfono. La vinoteca actúa como intermediario entre varias bodegas y los abonados, proporcionándoles las botellas de vino que éstos solicitan a precios con descuento.

La empresa tiene varios tipos de empleados: unos se encargan del almacén, de la actualización de la lista de vinos disponibles y los precios por botella y/o caja, otros se encarga de la contabilidad y otros de la atención al cliente así como de la gestión de los pedidos (y de dar de alta a los nuevos abonados). Se desea disponer de una aplicación que permita informatizar el trabajo de la empresa. Además, la empresa desea utilizar Internet como canal de ventas a los abonados.

Esta fue la descripción inicial del problema obtenida de las entrevistas realizadas en la empresa:

El encargado de las actualizaciones, según le llega la información de las bodegas, crea o modifica los vinos de cada bodega (indicando el nombre comercial del vino, el año, denominación de origen registrada, categoría –vino del año, crianza o reserva- y un breve comentario) y, para cada vino, las referencias disponibles (el código, el tipo de embotellado –por botellas o cajas-, el contenido en litros, el precio de la referencia y su disponibilidad). Las referencias se modifican cuando cambia el precio o se agotan pero no se borran del sistema. Además, suministra trimestralmente a los abonados un boletín personalizado con los vinos y referencias disponibles en ese momento y que coincidan con sus preferencias (expresadas en el momento de la inscripción).

Actualmente, los abonados pueden realizar sus pedidos en la tienda, por teléfono o correo electrónico al personal de atención al cliente. En la petición se indican los siguientes datos: nº de abonado, código de cada referencia y unidades solicitadas. Previamente a la aceptación del pedido, el personal de atención al cliente verifica que el abonado no tiene vencido ningún plazo de pago de pedidos anteriores (el límite para el pago es de 30 días una vez facturado), en cuyo caso el pedido se rechaza. Si todo es correcto se asigna un número de pedido que pasa a la situación de "pendiente". El rechazo o la confirmación del pedido se comunican al abonado por el mismo medio que éste utilizó para hacer el pedido.

El personal de almacén utiliza una vez a la semana los pedidos pendientes para generar una compra a cada bodega (nombre, CIF y dirección de la bodega más una relación de las referencias solicitadas y el número total de unidades de cada referencia). El pedido pasa al estado "tramitado". El personal de almacén se encarga también de introducir en el sistema las compras que llegan de las bodegas (la compra pasa del estado "pendiente" a "recibida") y asignar botellas y/o cajas a los pedidos tramitados. Cuando se han recibido todas las botellas y/o cajas de vino de un pedido tramitado, éste pasa automáticamente al estado "completado". El personal de atención al cliente se asegura de que los pedidos completados se remitan a los destinatarios a su dirección postal junto con la nota de entrega, una copia de la cual se envía al personal de contabilidad (y el pedido pasa a estado "servido").

Mensualmente el personal de contabilidad, con los pedidos servidos durante el mes elabora una

factura por abonado (evidentemente el cliente de la factura y el cliente de cada pedido debe ser el mismo) con los siguientes datos: el número y fecha de factura, la relación de los pedidos, el importe de cada uno, el importe total de estos pedidos, el banco y el número de cuenta corriente del abonado a la que se cargara el importe de la factura (y el pedido pasa a estado "facturado"). Con los extractos bancarios el personal de contabilidad da por pagadas las facturas y los pedidos pasan al estado "pagado".

Para poner en marcha el nuevo sistema, se pedirá al nuevo abonado el NIF, apellidos y nombre, dirección postal, dirección electrónica y cuenta corriente para domiciliación de los pagos y el sistema asignará automáticamente el nº de abonado. Además el abonado marcará sus preferencias (denominaciones de origen y categorías).

En el caso de los pedidos online, los abonados se identificarán y podrán consultar las referencias disponibles, según sus preferencias de categoría y denominación, indicando las que desee pedir y la cantidad (se irán añadiendo al pedido automáticamente). Cuando el abonado indique que ha terminado de seleccionar las referencias, si no tiene pendiente ningún plazo de pago se guardará el pedido como "pendiente" y se enviará por correo electrónico la confirmación del mismo. La confirmación incluirá los datos del abonado, del pedido (nº y fecha) y de las botellas o cajas de vino solicitadas (nombre comercial, año, categoría, bodega, nº de unidades y precio).

Se han realizado las fases de Elicitación y Especificación de Requisitos, se han descrito los requisitos funcionales, no funcionales y los casos de uso y de Análisis, cuya documentación parcial se aporta a continuación. Se presenta el Modelo del Dominio, un fragmento del diagrama de casos de uso y la especificación de cuatro de los casos de uso. Se aporta también el esquema de la base de datos realizado en la fase de Diseño a partir del Modelo Conceptual expresado en el Modelo de Dominio.

Se ha decidido separar el sistema en dos subsistemas, el que será desarrollado para el uso de los empleados de la empresa y el que será desarrollado para los abonados. El subsistema dedicado a los abonados será una aplicación web y no será abordada en esta asignatura. Se está estudiando aún la forma en la que se identificarán los abonados ya que puede permitirse identificación a través de sus redes sociales favoritas o mediante cualquier proveedor de identidades OpenID.

1.1.1. Restricciones adicionales

El sistema deberá utilizar una base de datos (BD), cuyo diseño se aporta, implementada con Derby.

Deberá asegurarse que todos los usuarios están previamente identificados en el sistema para acceder a cualquier función, y que las funciones serán las que correspondan al usuario según su rol en la empresa.

...

Se ha decidido realizar una aplicación de escritorio con acceso a BD. En teoría la BD sería centralizada en modo cliente-servidor, pero para facilitar las prácticas en este caso supondremos la conexión a una BD en local. Diseñado e implementado de forma que sea inmediato realizar los cambios necesarios para que la BD pase a ser remota y centralizada.

Se han realizado las fases de Elicitación y Especificación de Requisitos. Se han descrito los requisitos funcionales, no funcionales, de interacción y de información.

A partir de los requisitos, en la fase de análisis se ha realizado el Modelado del Dominio y la especificación de casos de uso en análisis. A continuación se aporta una documentación parcial de análisis que incluye: el Modelo del Dominio, un fragmento del diagrama de casos de uso y la especificación de cuatro de los casos de uso.

Se aporta también el diseño de la base de datos (diseño lógico y físico), realizado en la primera fase del Diseño a partir del modelado conceptual expresado en el Modelo de Dominio.

Capítulo 2

Documentación parcial de Análisis

2.1. Vista parcial del diagrama de Casos de Uso

Se aporta una vista parcial del diagrama de Casos de Uso. Para esta práctica se han escogido aquellos que se encuentran resaltados y para los que se adjunta una especificación.

Se han definido distintos actores. Los que vamos a considerar en este trabajo son:

- el actor generalización Empleado
- el actor Personal de atención al cliente
- el actor Personal de contabilidad
- el actor Personal de almacén

En la Figura 2.1 se muestra el mencionado fragmento del diagrama de casos de uso. Tal como se aprecia (resaltados) en dicha figura, para este trabajo vamos a tener en cuenta los siguientes casos de uso:

Empleado CU Identificarse

Personal de atención al cliente CU Crear pedido de abonado

Personal de contabilidad CU Consultar impagos

Personal de almacén CU Registrar recepción de compra

Aclaraciones sobre el caso de uso "Identificarse": El empleado se identifica con su DNI y contraseña. Si todo va bien, se carga su perfil con las opciones que puede realizar este tipo de empleado.

Para esta práctica, a cada tipo de actor se le mostrará una lista de opciones. Esta lista debe contener al menos el CU que se pide realizar para ese actor en este supuesto práctico. Para listar el resto de opciones se utilizarán los otros ejemplos de CU que se muestran en la Figura 2.1. La funcionalidad asociada a estas opciones será mostrar un mensaje que informe de "Opción aún no implementada".

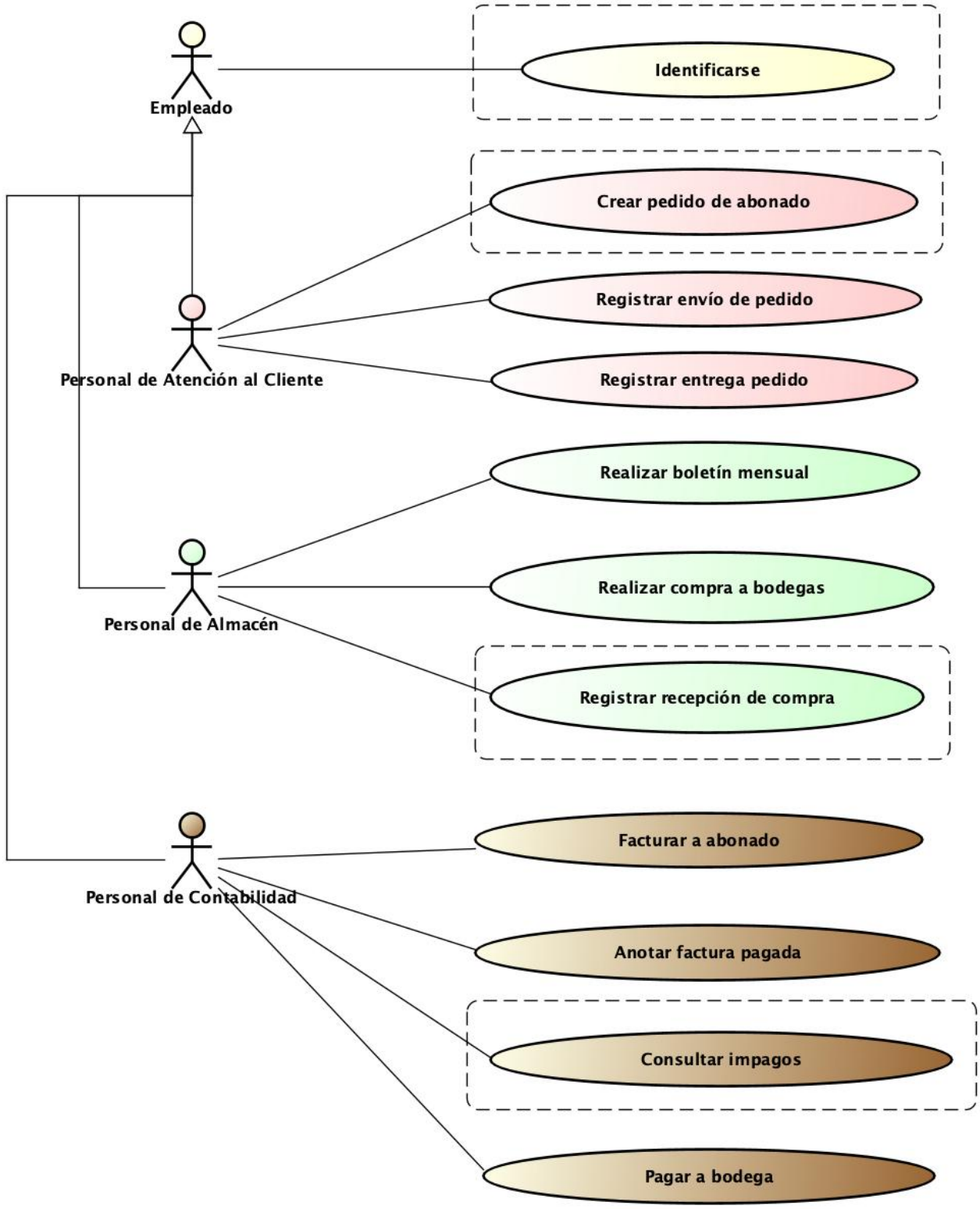

Figura 2.1: Fragmento del Diagrama de casos de uso. Se muestran sólo algunos ejemplos por actor.

2.2. Modelo del dominio

En la fase de análisis se ha obtenido el modelo del dominio que se muestra en la Figura 2.2. En dicho diagrama se ha omitido a propósito la especificación de restricciones OCL que sí son objetivo de la asignatura de Modelado de Sistemas Software. Nótese que no se han incorporado en este modelo de dominio algunos atributos de información y clases. El objetivo es simplificar para tener una práctica más asequible.

En los siguientes apartados se describen los escenarios de los casos de uso mencionados.

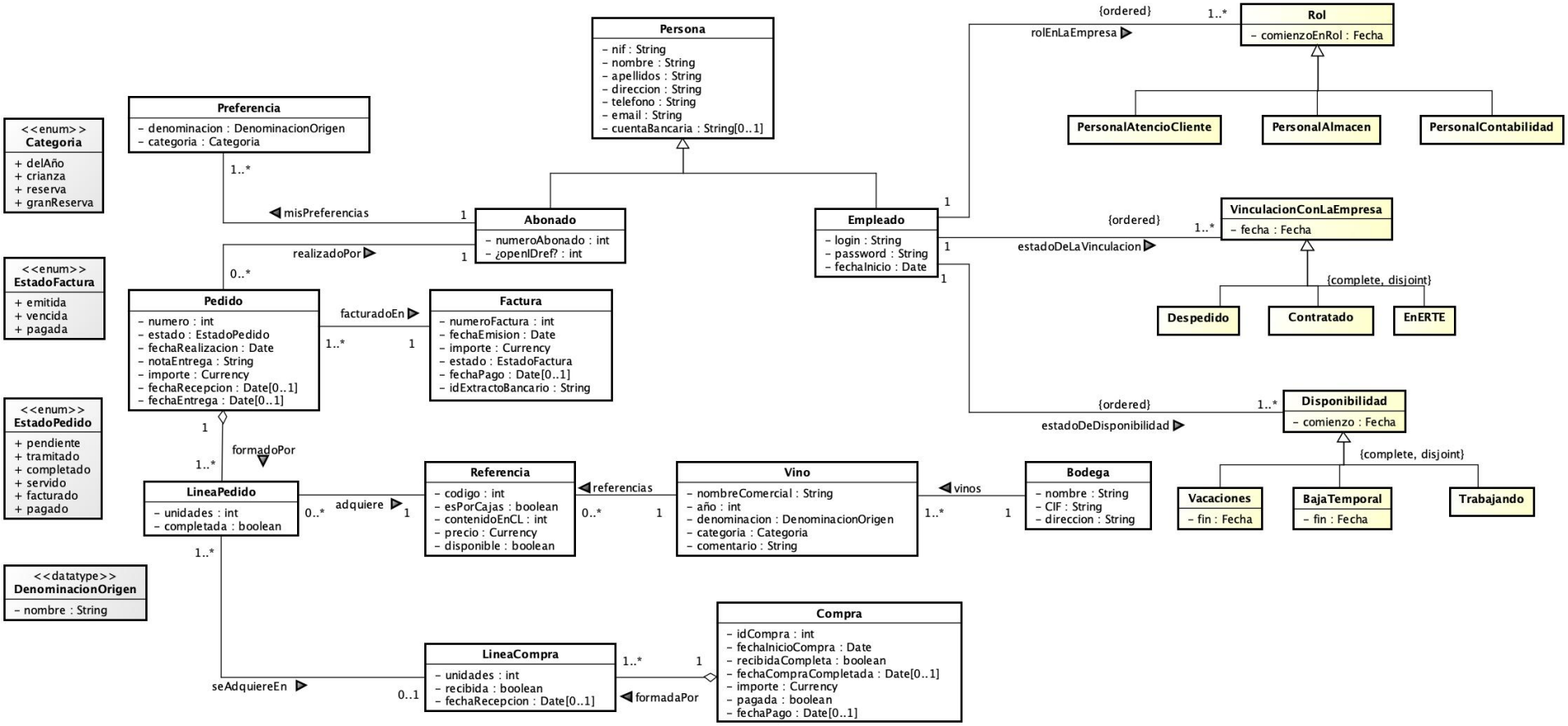

Figura 2.2: Modelo del Dominio. Punto de partida del diagrama de clases inicial

2.3. Especificación de casos de uso

De la misma forma que se aclaró en el apartado relativo al Modelo del Dominio, la especificación de casos de uso que aquí se describe no se corresponde exactamente con lo esperado con un caso real. Se han realizado muchas simplificaciones para que no sea necesario trabajar con demasiadas clases de dominio ni tablas en la base de datos.

2.3.1. Empleado: Identificarse

Actor: Empleado

Caso de Uso: Identificarse

Precondición: El actor no se encuentra identificado en el sistema

Secuencia normal:

1. El actor Empleado introduce dni y contraseña.
2. El sistema comprueba que dicho par dni-contraseña corresponden con un empleado en activo y muestra las opciones correspondientes a su rol en el sistema.

Alternativas y excepciones:

(2 a) Si no existe empleado con dni indicado, el sistema muestra un mensaje de error, a continuación el caso de uso queda sin efecto.

(2 b) Si la contraseña es incorrecta para el empleado con el dni indicado, el sistema muestra un mensaje de error, a continuación el caso de uso queda sin efecto.

(2 c) Si el empleado no está en activo (activo es: contratado y trabajando), el sistema muestra un mensaje de error y a continuación el caso de uso queda sin efecto.

Postcondición: El empleado está identificado en el sistema y sus opciones de trabajo mostradas.

2.3.2. Contabilidad: Consultar impagos

Actor: Personal de Contabilidad

Caso de Uso: Consultar impagos

Precondición: El actor se encuentra identificado en el sistema

Secuencia normal:

1. El actor indica una fecha para consultar todas las facturas anteriores a esa fecha de emisión que están vencidas
2. El sistema comprueba que se trata de una fecha anterior a 30 días de la fecha actual.
3. El sistema muestra al usuario un informe con todas las facturas emitidas antes de la fecha que están vencidas con el dato de los pedidos que forman dicha factura y el abonado responsable.

Alternativas y excepciones:

(2) El sistema comprueba que se trata de una fecha de emisión en los últimos 30 días e informa al usuario que las facturas en los últimos 30 días aún no están vencidas, a continuación el caso de uso continúa por el paso 3.

Postcondición: Se ha producido un informe para el usuario pero no se ha modificado el estado del sistema.

Indicaciones de usabilidad: el actor podrá elegir que los resultados se muestren ordenados según la fecha de la factura o según el abonado (cada criterio podrá ser elegido en primer o segundo lugar).

2.3.3. Personal de almacén: Registrar recepción de compra

Actor: Personal de Almacén

Caso de Uso: Registrar recepción de compra

Precondición: El actor se encuentra identificado en el sistema

Secuencia normal:

1. El actor introduce el identificador de la compra recibida
2. El sistema comprueba que se trata del identificador de una compra aún no completada y muestra los datos de la compra, en particular nombre de la bodega, y las líneas de compra aún no recibidas con sus datos de la referencia y el número de unidades
3. El actor selecciona la que se corresponde con lo recibido, comprobando que tiene todas las unidades
4. El sistema pasa la línea de compra a recibida, registra la fecha de recepción y revisa las líneas de pedido correspondientes indicándolas como completa.
5. El actor indica que ha finalizado de introducir líneas de compra recibidas
6. El sistema comprueba que todas las líneas de compras correspondientes a la compra en curso han sido recibidas, indica que la compra está recibidaCompleta y anota la fecha actual como fecha de compra completada
7. El sistema revisa los pedidos en estado tramitado comprobando si ya tienen todas sus líneas completas en cuyo caso lo pasa al estado "completado".

Alternativas y excepciones:

(2) El sistema no encuentra una compra con dicho identificador, muestra un mensaje, a continuación el caso de uso finaliza

(2) El sistema comprueba que el identificador introducido es de una compra ya completada, muestra un mensaje, a continuación el caso de uso finaliza

(5) El actor indica que no ha finalizado aún de introducir líneas de compra recibidas, a continuación el caso de uso continúa por el paso 3

(6) El sistema encuentra que algunas líneas de compra no han sido recibidas correctamente en cuyo caso informa al actor de las líneas de compras pendientes de completar para que pueda realizar una reclamación a la bodega, a continuación el caso de uso continúa por el paso 7

(3, 5) El actor cancela, a continuación el caso de uso queda sin efecto

Postcondición: Algunas líneas de compra pasan estar completas, así como algunas líneas de pedido, algunas compras pasan estar recibidaCompleta, algunos pedidos pasan al estado "completado"

2.3.4. Personal de Atención al Cliente: Crear pedido de abonado

Comentario: Si bien en el sistema se añadirá un subsistema para que los abonados puedan realizar y gestionar sus pedidos online, la empresa seguirá recibiendo pedidos de aquellos abonados que no deseen utilizar ese sistema online y lo soliciten personalmente en tienda, por teléfono o por email.

Actor: Personal de atención al cliente

Caso de Uso: Crear pedido de abonado

Precondición: El actor se encuentra identificado en el sistema

Secuencia normal:

1. El actor introduce el número del abonado
2. El sistema comprueba que existe un abonado con dicho número, muestra sus datos nombre, apellidos, teléfono, email y solicita confirmación de que se trata del abonado correcto.
3. El actor confirma
4. El sistema comprueba que el abonado no tiene vencido ningún plazo de pago de pedidos anteriores y crea un nuevo pedido
5. El actor introduce una referencia y la cantidad a adquirir de dicha referencia en el pedido
6. El sistema comprueba que se trata de una referencia válida y que la referencia está disponible, crea una línea de pedido y en el pedido en curso
7.) El actor indica que ya ha finalizado de introducir el pedido
8. El sistema registra el nuevo pedido

Alternativas y excepciones:

(1) Si el actor lo elige se ejecuta el caso de uso Buscar Abonado

(3) El actor indica que es un error, a continuación el caso de uso continúa por el paso 1

(4) El sistema encuentra que el abonado tiene vencido algún plazo de pago de pedidos anteriores, notifica con un mensaje de error y produce un texto para facilitar el envío de una notificación al abonado si fuera el caso de pedido por correo electrónico, a continuación el caso de uso queda sin efecto.

(6) El sistema comprueba que no existe esa referencia, muestra el error y a continuación el caso de uso continúa por el paso 5.

(6) El sistema comprueba que la referencia no está disponible, muestra el mensaje de error y a continuación el caso de uso continúa por el paso 5.

(7) El actor indica que no ha finalizado aún de introducir el pedido, el caso de uso continúa por el paso 5.

(3, 5, 7) El actor cancela, a continuación el caso de uso queda sin efecto.

Postcondición: Escenario de éxito: Se ha creado un nuevo pedido en estado "pendiente" y tantas líneas de pedido (no completadas) como referencias formen parte de este.

Simplificaciones a la usabilidad: Para simplificar esta práctica no se tienen en cuenta modificaciones durante el pedido, retirar una línea de perdido, modificar una cantidad, una referencia, etc. Tampoco se tiene en cuenta que se puede leer el código del producto mediante dispositivos lectores de código de barras, etc.

2.4. Restricciones a tener en cuenta en la implementación

El sistema deberá utilizar una base de datos cuyo esquema (diseño lógico) se aporta en la página 14, implementada con Derby. Deberá utilizarse el script de creación de la base de datos (diseño físico) se aporta en la página 15. El sistema será una aplicación de escritorio independiente del sistema operativo que se desarrollará en JAVA, jdk 11 utilizando Swing y JDBC. El sistema será desarrollado en NETBEANS en la misma versión que se encuentra instalada actualmente en los laboratorios de la Escuela (actualmente Netbeans 11.2).

2.5. Análisis arquitectónico

En el subsistema para el trabajo interno de la empresa se ha decidido aplicar una arquitectura de capas con tres capas estricta y una relajada de servicios útiles a todas las capas, MVC y DAO+DTO como patrón de acceso a datos.

Capítulo 3

Diseño de Datos

3.1. Diseño lógico (datamodel style)

En la Figura 3.1 se muestra el diseño de los datos como esquema relacional (diseño lógico), utilizando una representación basada en UML con estereotipos. Los estereotipos «PK» y «FK» representan claves primarias y claves foráneas, respectivamente. Los clasificadores con el estereotipo «table» representan una tabla en el esquema relacional. Las tablas que representan un elemento del dominio se identifican con el estereotipo «entity». Las tablas que representa asociaciones se identifican con el estereotipo «association». Las tablas que representan valores enumerados se identifican con el estereotipo «enum» y aparecen sombreadas. La transformación al modelo relacional de la relación de herencia (Persona, Empleado, Abonado) presente en el Modelo del Dominio se ha realizado mediante referencias entre tablas. El modelado de roles temporales presentes en el Modelo del Dominio para Empleado se ha realizado mediante tablas asociación y tablas enum.

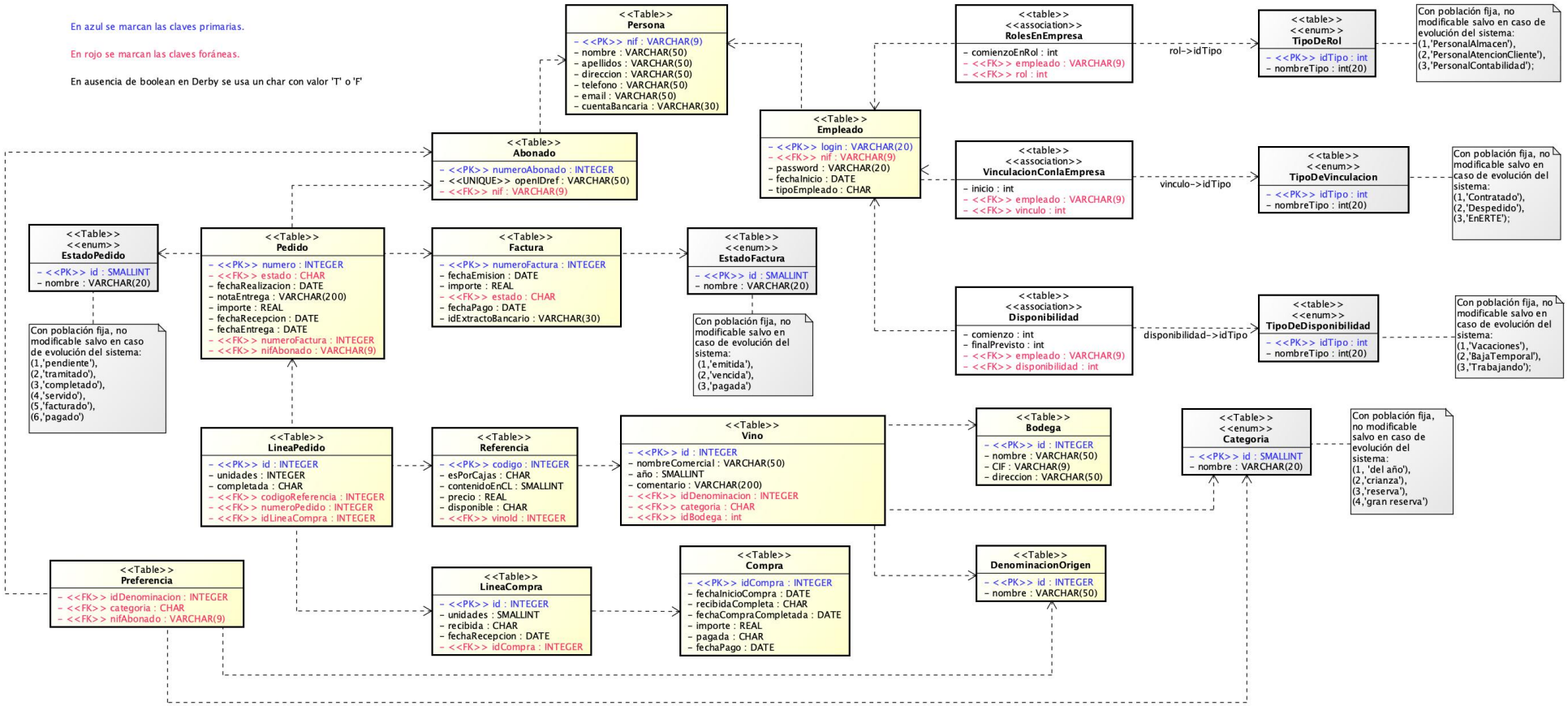

Figura 3.1: Diseño lógico de la base de datos relacional.

3.2. Scripts de creación de la base de datos

Este script puede obtenerse en un archivo almacenado como recurso en el aula virtual siguiendo el enlace: `https://aulas.inf.uva.es/pluginfile.php/56202/mod_resource/content/1/createTables.sql`.

```
--Derby does not support DROP TABLE IF EXISTS
DROP TABLE DISPONIBILIDADEMPLEADO;
DROP TABLE VINCULACIONCONLAEMPRESA;
DROP TABLE ROLESENEMPRESA;
DROP TABLE TIPODEDISPONIBILIDAD;
DROP TABLE TIPODEVINCULACION;
DROP TABLE TIPODEROL;
DROP TABLE LINEAPEDIDO;
DROP TABLE PEDIDO;
DROP TABLE LINEACOMPRA;
DROP TABLE COMPRA;
DROP TABLE ESTADOPEDIDO;
DROP TABLE FACTURA;
DROP TABLE ESTADOFACTURA;
DROP TABLE PREFERENCIA;
DROP TABLE REFERENCIA;
DROP TABLE VINO;
DROP TABLE BODEGA;
DROP TABLE CATEGORIA;
DROP TABLE DENOMINACIONORIGEN;
DROP TABLE ABONADO;
DROP TABLE EMPLEADO;
DROP TABLE PERSONA;

-- Enum
create table TIPODEROL
(
        IdTipo SMALLINT not null,
        NombreTipo VARCHAR(20) not null unique,
                PRIMARY KEY(IdTipo)
);

INSERT INTO TIPODEROL
VALUES  (1,'Almacen'),
        (2,'AtencionCliente'),
        (3,'Contabilidad');

-- Enum
create table TIPODEVINCULACION
(
        IdTipo SMALLINT not null,
        NombreTipo VARCHAR(20) not null unique,
                PRIMARY KEY(IdTipo)

);

INSERT INTO TIPODEVINCULACION
VALUES  (1,'Contratado'),
        (2,'Despedido'),
        (3,'EnERTE');

-- Enum
create table TIPODEDISPONIBILIDAD
(
        IdTipo SMALLINT not null,
        NombreTipo VARCHAR(20) not null unique,
                PRIMARY KEY(IdTipo)
);
```

```
INSERT INTO TIPODEDISPONIBILIDAD
VALUES  (1,'Vacaciones'),
        (2,'BajaTemporal'),
                (3, 'Trabajando');

-- Entity
CREATE TABLE PERSONA(
        Nif VARCHAR(9),
        Nombre VARCHAR(50) not null,
        Apellidos VARCHAR(50) not null,
        Direccion VARCHAR(50) not null,
        Telefono VARCHAR(50) not null,
        Email VARCHAR(50) not null,
        CuentaBancaria VARCHAR(30) not null,
        PRIMARY KEY(Nif)
);

-- Entity
create table EMPLEADO
(
        Nif VARCHAR(9),
        Password VARCHAR(20) not null,
        FechaInicioEnEmpresa DATE not null,
                PRIMARY KEY(Nif),
                FOREIGN KEY(Nif) REFERENCES PERSONA(Nif)
);

-- Association
create table ROLESENEMPRESA
(
        ComienzoEnRol DATE not null,
        Empleado VARCHAR(9) not null,
        Rol SMALLINT not null,
            FOREIGN KEY(Empleado) REFERENCES EMPLEADO(Nif),
            FOREIGN KEY(Rol) REFERENCES TIPODEROL(IdTipo)
);

-- Association
create table VINCULACIONCONLAEMPRESA
(
        inicio DATE not null,
        Empleado VARCHAR(9) not null,
        Vinculo SMALLINT not null,
                FOREIGN KEY(Empleado) REFERENCES EMPLEADO(Nif),
                FOREIGN KEY(Vinculo) REFERENCES TIPODEVINCULACION(IdTipo)
);

-- Association
create table DISPONIBILIDADEMPLEADO
(
        Comienzo DATE not null,
        FinalPrevisto DATE,
        Empleado VARCHAR(9) not null,
        Disponibilidad SMALLINT not null,
                FOREIGN KEY(Empleado) REFERENCES EMPLEADO(Nif),
                FOREIGN KEY(Disponibilidad) REFERENCES TIPODEDISPONIBILIDAD(IdTipo)
);

-- Entity
CREATE TABLE ABONADO(
        NumeroAbonado INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT
            BY 1),
        OpenIDref VARCHAR(50) unique,
        Nif VARCHAR(9) not null unique,
        PRIMARY KEY(NumeroAbonado),
        FOREIGN KEY(Nif) REFERENCES PERSONA(Nif)
```

```
);

-- Enum
CREATE TABLE ESTADOFACTURA(
        Id SMALLINT,
        Nombre VARCHAR(20) not null unique,
        PRIMARY KEY(id)
);

INSERT INTO ESTADOFACTURA
VALUES  (1,'emitida'),
                (2,'vencida'),
                (3,'pagada');

-- Entity
CREATE TABLE FACTURA(
        NumeroFactura INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT
            BY 1),
        FechaEmision DATE,
        Importe REAL,
        Estado SMALLINT,
        FechaPago DATE,
        IdExtractoBancario VARCHAR(30),
        PRIMARY KEY(NumeroFactura),
        FOREIGN KEY(Estado) REFERENCES ESTADOFACTURA(Id)
);

-- Enum
CREATE TABLE CATEGORIA(
        Id SMALLINT,
        Nombre VARCHAR(20) not null unique,
        PRIMARY KEY(Id)
);

INSERT INTO CATEGORIA
VALUES  (1,'joven'),
                (2,'crianza'),
                (3,'reserva'),
                (4,'gran reserva');

-- Entity
CREATE TABLE DENOMINACIONORIGEN(
        Id INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
        Nombre VARCHAR(50) not null unique,
        PRIMARY KEY(Id)
);

-- Entity
CREATE TABLE PREFERENCIA(
        IdDenominacion INTEGER,
        Categoria SMALLINT,
        NumeroAbonado INTEGER,
        FOREIGN KEY(IdDenominacion) REFERENCES DENOMINACIONORIGEN(Id),
        FOREIGN KEY(Categoria) REFERENCES CATEGORIA(Id),
        FOREIGN KEY(NumeroAbonado) REFERENCES ABONADO(NumeroAbonado)
);

-- Entity
CREATE TABLE BODEGA(
        Id INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
        Nombre VARCHAR(50) not null unique,
        CIF VARCHAR(9) not null unique,
        Direccion VARCHAR(50) not null,
        PRIMARY KEY(Id)
);
```

```sql
-- Entity
CREATE TABLE VINO(
        Id INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
        NombreComercial VARCHAR(50) not null unique,
        Ano SMALLINT,
        Comentario VARCHAR(200),
        IdDenominacion INTEGER,
        Categoria SMALLINT,
        IdBodega INTEGER,
        PRIMARY KEY(Id),
        FOREIGN KEY(IdDenominacion) REFERENCES DENOMINACIONORIGEN(Id),
        FOREIGN KEY(Categoria) REFERENCES CATEGORIA(Id),
        FOREIGN KEY(IdBodega) REFERENCES BODEGA(Id)
);

-- Entity
CREATE TABLE REFERENCIA(
        Codigo INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
        EsPorCajas CHAR(1),
        ContenidoEnCL SMALLINT,
        Precio REAL,
        Disponible CHAR(1),
        VinoId INTEGER,
        PRIMARY KEY(Codigo),
        FOREIGN KEY(VinoId) REFERENCES VINO(Id)
);

-- Entity
CREATE TABLE COMPRA(
        IdCompra INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1)
            ,
        FechaInicioCompra DATE,
        RecibidaCompleta CHAR(1),
        FechaCompraCompletada DATE,
        Importe REAL,
        Pagada CHAR(1),
        FechaPago DATE,
        PRIMARY KEY(IdCompra)
);

-- Entity
CREATE TABLE LINEACOMPRA(
        Id INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
        Unidades SMALLINT,
        Recibida CHAR(1),
        FechaRecepcion DATE,
        IdCompra INTEGER,
        PRIMARY KEY(Id),
        FOREIGN KEY(IdCompra) REFERENCES COMPRA(IdCompra)
);

-- Enum
CREATE TABLE ESTADOPEDIDO(
        Id SMALLINT,
        Nombre VARCHAR(20)not null unique,
        PRIMARY KEY(Id)
);

INSERT INTO ESTADOPEDIDO
VALUES (1,'pendiente'),
                (2,'tramitado'),
                (3,'completado'),
                (4,'servido'),
                (5,'facturado'),
                (6,'pagado');
```

```sql
-- Entity
CREATE TABLE PEDIDO(
        Numero INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
        Estado SMALLINT,
        FechaRealizacion DATE,
        NotaEntrega VARCHAR(200),
        Importe REAL,
        FechaRecepcion DATE,
        FechaEntrega DATE,
        NumeroFactura INTEGER,
        NumeroAbonado INTEGER,
        PRIMARY KEY(Numero),
        FOREIGN KEY(Estado) REFERENCES ESTADOPEDIDO(Id),
        FOREIGN KEY(NumeroFactura) REFERENCES FACTURA(NumeroFactura),
        FOREIGN KEY(NumeroAbonado) REFERENCES ABONADO(NumeroAbonado)
);

-- Entity
CREATE TABLE LINEAPEDIDO(
        Id INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
        Unidades INTEGER,
        Completada CHAR(1),
        CodigoReferencia INTEGER,
        NumeroPedido INTEGER,
        IdLineaCompra INTEGER,
        PRIMARY KEY(Id),
        FOREIGN KEY(CodigoReferencia) REFERENCES REFERENCIA(Codigo),
        FOREIGN KEY(NumeroPedido) REFERENCES PEDIDO(Numero),
        FOREIGN KEY(IdLineaCompra) REFERENCES LINEACOMPRA(Id)
);
```

Capítulo 4

Normas de entrega y criterios de evaluación

4.1. Indicaciones para la entrega

El seguimiento del proyecto se realizará a través de la aplicación Pivotal Tracker: `https://www.pivotaltracker.com`.

Todos los alumnos se habrán creado una cuenta en Pivotal Tracker y habrán sido añadidos al proyecto correspondiente según su equipo de trabajo. El alumno deberá estar identificado con su correo en alumnos.uva.es y su identificación como login en los laboratorios de la Escuela.

La comunicación con los equipos puede hacer mediante Pivotal Tracker o mediante el canal de comunicación en la instancia rocket de la Escuela:
`https://rocket.inf.uva.es`

Para realizar la entrega se preparará una *release* en Pivotal Tracker. En dicha *release* se adjuntará un archivo zip o tgz (o tar.gz) con el contenido que se especifica en el apartado 4.2.

Se deja a decisión de los alumnos miembros de un equipo, que la entrega correspondiente a la *release* se realice mediante un repositorio en el GitLab de la Escuela (cumpliendo estrictamente la estructura y contenidos que se especifican en el apartado 4.2). Esto último es opcional ya que no forma parte de los contenidos de esta asignatura pero puede ser preferencia del alumnado que ya domina estas tecnologías trabajar de esta forma. Para esto último deberán seguirse las siguientes normas:

- La entrega se realizará añadiendo a la profesora (usuario `yania`) con permisos de tipo `Reporter` al repositorio que contiene el proyecto en el GitLab de la Escuela cuando ya no se vaya a realizar ningún *commit+push*. Cualquier *push* al repositorio una vez vencido el plazo de entrega será penalizado con 0 en la Práctica en la convocatoria correspondiente.
- El enlace (url) al repositorio y cualquier documentación necesaria al respecto serán anotadas en la *release* en Pivotal Tracker.

4.2. Estructura y contenidos de la entrega

La entrega tendrá la siguiente estructura de carpetas:

```
models/
app/
```

En la carpeta `models` se encontrará un archivo Astah Professional o Visual Paradigm (a elección del equipo). Los modelos y diagramas contenidos en dicho archivo Astah Professional o Visual Paradigm que son evaluables corresponden a lo realizado en la fase de diseño. Se tendrá cuidado de alojar todo lo realizado en un modelo llamado Diseño organizado con sus respectivos submodelos y diagramas.

Se contará con los diagramas necesarios para representar la arquitectura de referencia, los estilos universales *decomposition style*, *uses styl*e, *inheritance style* y *data model* así como los diagramas de

clases de diseño detallado y los diagramas de secuencia con la realización en diseño de los casos de uso que se piden en este enunciado. Adicionalmente, se aportará un diagrama de estados para modelar la interfaz de usuario del sistema (en el que se modelará únicamente lo necesario para los casos de uso que se especifican en este documento). Los diagramas tendrán que ser legibles y comprensibles. Si los diagramas se hacen excesivamente grandes deberán utilizarse los elementos que ofrece UML para reducir el tamaño y la complejidad de los modelos.

En la carpeta `app` se espera una estructura como la siguiente:

```
app/netbeansProject/
app/db/
```

La carpeta `app/netbeansProject` contendrá, como su nombre indica, el proyecto NETBEANS que implementa el caso de uso. La implementación no debe incumplir el diseño propuesto.

La carpeta `app/db` contendrá un archivo config.db de propiedades con lo necesario para conectarse con la BD. Este archivo indicará la url con el puerto, el nombre de la base de datos, el usuario y el password exactamente como se indica a continuación:

```
url=jdbc:derby://localhost:1527/vinoteca
user=adminvinoteca
password=vinoteca2020
```

En dicha carpeta (`app/db`) también se encontrarán unos scripts que permitirán la regeneración de la base de datos (el suministrado con este enunciado para crear las tablas) así como tantos scripts SQL como sean necesarios para poblar automáticamente la base de datos de cara a probar la aplicación (puede ser uno o varios separados por casos de uso o escenarios) siempre que se documente apropiadamente el propósito de cada uno.

Ni en el control de versiones, ni en los archivos de entrega debe residir la base de datos Derby.

Fecha límite para la entrega

A continuación se especifican las **fechas límite de entrega para cada convocatoria**. Esta será la fecha de la *release* en PIVOTAL TRACKER. Si no se cumple la fecha límite, el equipo será penalizado con 0 en la Práctica en la convocatoria correspondiente.

convocatoria ordinaria: 31 de mayo de 2020

convocatoria extraordinaria: 21 de junio de 2020

SITUACIÓN ESPECIAL: Debido a la incidencia ocasionada por la pandemia Covid19, estas fechas especificadas en el enunciado no se considerarán definitivas. A la espera de la decisión de las autoridades académicas de la Universidad de Valladolid, si así se decidiera, se moverán tanto como se decida mover el calendario académico. Se ha destinado un 5 % de la nota para premiar a aquellos que vayan haciendo un seguimiento de la práctica en la *releases* intermedias.

4.3. Criterios de Evaluación

Se valorará:

(a) la aplicación y consistencia en el diseño de la arquitectura de 3 capas (capas estrictas) combinada con MVC, se puede considerar además añadir una capa transversal de servicios (capa relajada);

(b) la aplicación de los patrones GRASP y algunos patrones de diseño conocidos, en particular la aplicación de patrones de acceso a datos;

(c) la corrección y completitud de los modelos UML.

(d) la calidad de la solución.

Los criterios anteriores tendrán el mayor peso en la evaluación de la práctica (70 %) y se desglosan de la siguiente forma:

- diagramas de la arquitectura de referencia y descomposición modular- 0,5/10
- diagramas de dependencias entre capas - 0,4/10
  - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Identificarse? 0,1/10
  - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Crear pedido de abonado? - 0,1/10
  - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Registrar recepción de compra? - 0,1/10
  - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Consultar impagos? - 0,1/10
- diagramas de clases de diseño detallado - 0,5/10
- diagrama de estados que modela la interfaz - 0,1/10
- diagramas de secuencia con la Realización en Diseño de los CU-
  - ¿Se diseña adecuadamente la realización del CU Identificarse (incluye mecanismo de persistencia)? - 1/10
  - ¿Se diseña adecuadamente la realización del CU Crear pedido de abonado (incluye mecanismo de persistencia)?- 1,5/10
  - ¿Se diseña adecuadamente la realización del CU Registrar recepción de compra (incluye mecanismo de persistencia)? - 1,5/10
  - ¿Se diseña adecuadamente la realización del CU Consultar impagos)? - 1,5/10

Respecto del código de la aplicación (30 %) se valorará:

(e) que sea consistente con el diseño arquitectónico (1/10);

(f) que sea consistente con el modelo dinámico diseñado (es decir, con los diagramas de secuencia que describen la realización en diseño de los casos de uso) (1/10);

(g) que se comporte según lo esperado en las situaciones válidas y que no acepte situaciones inválidas comunicando al usuario los errores que se controlan (pruebas de aceptación superadas) (1/10).

La calidad y la seguridad del código de la aplicación será analizada mediante SonarQube como se explica en el aula virtual (`https://aulas.inf.uva.es/mod/page/view.php?id=27169`).

Los proyectos con buenos indicadores de calidad y seguridad, detectados por esta herramienta, podrán optar a un premio.

El premio a repartir son puntos en la asignatura. Se otorgará 0,9, 0,6 y 0,3 como premio al primero, segundo, y tercer lugar en el concurso, respectivamente. No habrá empates en ninguna de las tres posiciones del concurso. Al sumar los premios obtenidos en la nota de los ganadores, no podrá sobrepasarse los 10 puntos como nota global de la asignatura.

Se elaborará un ranking entre los equipos participantes.

Para entrar en el ranking se requiere:

- pasar el QualityGate DS definido en el servidor sonarqube (marca verde Passed por el contrario de marca roja Failed). Este QualityGate se puede consultar en: `https://sonarqube.inf.uva.es/quality_gates/show/2`.
- haber implementado toda la funcionalidad especificada en los casos de uso de este enunciado.

Una vez dentro del ranking se establecerá un orden inverso (de mayor a menor puntuación, es decir, menos puntos mejor posición en el ranking), asignando una puntuación a cada equipo: $RDT + DD + VRE + BRE$

Donde RTD es la ratio de deuda técnica acumulada, DD es la densidad de duplicados, VRE es la tasa de esfuerzo para remediar las vulnerabilidades y BRE es la tasa de esfuerzo para remediar bugs potenciales.

Los empates se resolverán aplicando los siguientes criterios en este orden:

1. el que tenga menor ratio de deuda técnica,
2. el que tenga menor severidad de los code smells, es decir, el que tenga menor la ratio blocking+major)/violations
3. el que tenga menor porcentaje de código duplicado,
4. el que tenga menos bugs potenciales
5. el que tenga menos vulnerabilidades
6. finalmente el que tenga mayor porcentaje de comentarios javadoc (si bien no es evaluable en esta práctica, todos sabemos lo importante que es para la mantenibilidad).

La forma de uso de Pivotal Tracker no será evaluable, pero el uso será obligatorio. En caso de no utilizarse, la práctica no podrá ser considerada entregada ya que las *releases* parciales y la *release* final se realizan por esa vía.

El uso de rocket no será evaluable ni obligatorio, aunque es recomendable para introducir debates en los que pueda participar el profesor y aportar sugerencias o preguntar dudas que pueda resolver el profesor u otros compañeros, así como notificar de algún error u omisión detectado en el enunciado, los modelos, el script sql para la creación de la BD, etc..