Universitat de Vic
Escola Politècnica Superior

Treball de Final de Carrera

Programa de gestió per la Pastisseria Mas

Eduard Mateos Martínez

Enginyeria Tècnica d'Informàtica de Sistemes

Director: Jaume Vila Serra

Vic, Setembre de 2009

SUMARI

0. Resums

0.1. Resum en català

Eng. Tècnica d'informàtica de Sistemes

Títol: Programa de gestió per la Pastisseria Mas

Paraules clau: Gestió, C++, PHP,UML.

Autor: Eduard Mateos Martínez

Direcció: Jaume Vila Serra

Data: Setembre de 2009

Resum

En els últims anys, el món de la informàtica ha evolucionat d'una manera inimaginable, tan a nivell de Hardware com de Software. Aquesta evolució ha donat lloc a la creació de moltes empreses dedicades a la programació, on una de les seves principals feines ha estat la realització de programes de gestió d'empreses.

Moltes vegades, però, els programes estàndards no poden satisfer el total de les necessitats dels clients, sinó algunes d'aquestes i realitzar un programa personalitzat té un cost elevat.

En el cas de la *Pastisseria Mas* de Navàs*,* una empresa familiar, per poder realitzar les tasques administratives utilitzen fulls de càlcul, concretament el *Microsoft Excel*, que permet portar els comptes d'una manera més o menys senzilla, ja que només són necessàries unes nocions bàsiques d'informàtica. El mateix passa amb les dades dels proveïdors, que les guarden en una Base de Dades del tipus *Microsoft Access*. Una altra de les mancances és el tema dels encàrrecs que es fa de manera manual.

Per tant, l'objectiu d'aquest projecte, és realitzar un programa que els hi faciliti la seva activitat. Aquest programa els permetrà gestionar les dades que utilitzen, com la informació sobre els clients, personal, comandes... També s'ha desenvolupat una web que permet obtenir informació sobre les comandes que s'han realitzat.

Aquesta aplicació està dissenyada per funcionar en l'entorn *Windows XP* i s'ha desenvolupat amb el compilador de *CodeGear Rad Studio*, concretament el *C++ Builder 2009*. A nivell de base de dades, he utilitzat *MySQL* i en el cas de la pàgina web, *PHP* i la mateixa base de dades. L'anàlisi i el disseny ha estat fet en *UML*.

0.2. English Summary

Eng. Tècnica d'informàtica de Sistemes

Title: Programa de gestió per la Pastisseria Mas

Key words: manage program, C++, PHP,UML.

Author: Eduard Mateos Martínez

Director: Jaume Vila Serra

Date: Setembre de 2009

Summary

Nowadays, the computing world has experienced a huge development in Hardware as well as Software. So, a lot of programming companies have been set up. One of their main tasks is to create Software to manage companies.

However, a lot of times, the standardized programs can't satisfy the customer's necessities. What is more, buying a personalized program is too expensive for them.

In my project, *Pastisseria Mas* of Navàs, a familiar cake shop, to manage their company they use spreadsheeds, exactly the *Microsoft Excel* ones. It is a good option for them because they can handle their accounts easily with basic computing skills.
Similarly at that case, they keep the essential details in a *Microsoft Access* database. Another problem that face the company is the lack of a computerized system to admin the orders which are written down.

In conclusion, the aim of this project is creating a manage program that could facility the activities of the cake shop. This program will be able to handle customer's personal details, information about orders, products, employees…
Moreover, a web page will be developed to access easily to the information about the delivery notes of the database.

This application will be created to work with *Windows XP* and it is designed with the compilator *Red Studio* of *CodeGear*, specifically the *C++ Builder 2009*. The language used in the database is *MySQL* and *PHP* in the webpage. The analysis and the design are done in *UML*.

1. Proposta i estudi del problema

Ens interessa desenvolupar una aplicació que permeti gestionar els diferents elements d'una empresa alimentària familiar.

En aquesta empresa hi ha tres tipus de categories laborals: el propietari, els pastissers i els botiguers, cadascú pot accedir a diferents parts del programa i amb diferent nivell de privilegis. D'aquestes persones en sabem: el nom, els cognoms, el DNI, el NSS, el sou, l'adreça, el telèfon i el mòbil.

A la pastisseria es venen diferents productes, alguns tenen diferents tipus de farciment. Cada producte està compost d'unes determinades matèries primeres en diferent quantitat. Dels productes en sabem el nom, el preu i el tipus.

Les matèries primeres són distribuïdes pels proveïdors. Els diferents proveïdors poden tenir les mateixes matèries primeres però, com és normal, els preus són diferents per cadascun d'ells.
Dels proveïdors en sabem el nom, l'adreça, la localitat i un telèfon de contacte. De les matèries primeres en sabem el nom i el preu que tenen per cada proveïdor.

L'aplicació també ens ha de permetre controlar les comandes, que poden provenir dels clients o del personal de la pròpia pastisseria. De les comandes en coneixem la data d'entrega i l'estat en què es troba.
Dels clients en sabem el nom, els cognoms, la localitat, el telèfon i el mòbil.
El propietari vol consultar l'estat de les vendes a través de la web per consultar-ho a distància.
És interessant realitzar diferents utilitats en l'aplicació:

- Una utilitat que ens permeti calcular la quantitat de matèries primeres que es necessiten per poder elaborar les comandes que s'han de servir entre dues dates.
- Com ja hem comentat anteriorment, cada producte està format per diferents matèries primeres. Ens interessa poder calcular la quantitat de matèries

primeres que necessitem per fer una quantitat de producte final determinada.

- Gestionar el manteniment dels diferents objectes: comanda, personal, client, proveïdor, producte i matèria primera.

També cal tenir en compte, com ja hem comentat anteriorment, que els diferents tipus de personal tenen uns privilegis diferents:

- En el cas d'un pastisser: pot donar d'alta, de baixa i consultar els següents objectes: productes, farciments i receptes. També pot consultar les comandes i modificar l'estat d'aquestes.
- Els botiguers poden donar d'alta, de baixa i consultar els següents objectes: clients i comandes.
- El propietari pot donar d'alta, de baixa i consultar els objectes: personal, proveïdors i preu de les matèries primeres.

Amb tota aquesta informació observem que tenim un ***Sistema Discret d'Estat Únic (SDEU)*****,** ja que tots els objectes que trobem poden estar en un sol estat.
Per tant, utilitzarem l'estàndard **UML** per poder analitzar i dissenyar aquesta aplicació.

2. Especificació de requeriments

L'objectiu d'aquest apartat és saber quins són els elements que formaran part de la nostra aplicació i la interconnexió entre ells, així com les entitats participants que interactuaran amb el projecte informàtic.

Per tant, la finalitat és descriure de manera detallada el què ha de fer el projecte informàtic sense lligar-ho a cap tecnologia en concret.

2.1. Especificació del SDEU

Per poder descriure les accions que es duran a terme dins de l'empresa i les modificacions de l'objecte comanda, s'utilitzarà el **diagrama d'activitats**. Aquest ens permet veure com l'objecte es va modificant en funció del temps i les accions que realitzen les diferents persones per assolir un objectiu.

En la següent pàgina, s'utilitzarà per descriure el procés de comandes:

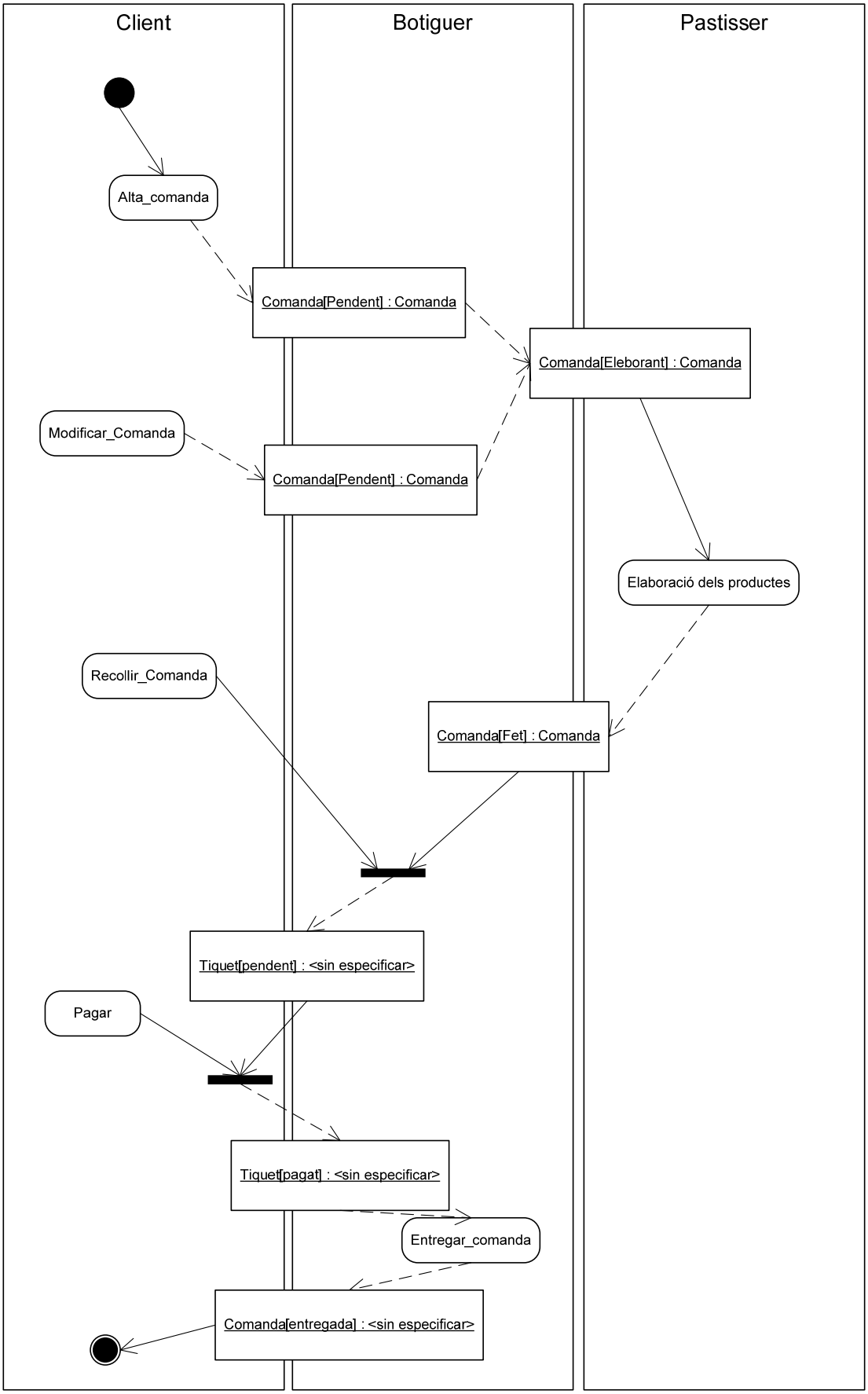


Diagrama d'activitat de comanda

La interpretació del diagrama és la següent:

Per realitzar una comanda, intervenen tres persones: el client, el botiguer i en últim lloc el pastisser. Per començar tot el procés, un client ha de sol·licitar una comanda, el botiguer és l'encarregat de formalitzar-la, és a dir, la comanda passa a estar **pendent**.
El client pot modificar la comanda si està en l'estat de **pendent**. Un cop formalitzada, el pastisser entra en joc i modifica l'estat de la comanda que passa a **elaborant** El pastisser, quan arriba el dia, procedeix a elaborar els productes sol·licitats i, un cop ha acabat, torna a modificar l'estat de la comanda que passa a **fet**. Quan el client va a recollir la comanda, aquesta ha de tenir l'estat **fet** per poder-li entregar. Un cop s'han complert aquestes dues condicions, es realitza el tiquet amb l'import de tota la comanda i, un cop ha pagat, se li entrega la comanda modificant l'estat a **entregat**.

2.1.1. Descripció del guió

En aquest apartat es descriurà la forma de treballar dels diferents usuaris respecte el *software* a desenvolupar.

El *software* només l'utilitzaran tres tipus diferents d'usuaris: el botiguer, el pastisser i el propietari.

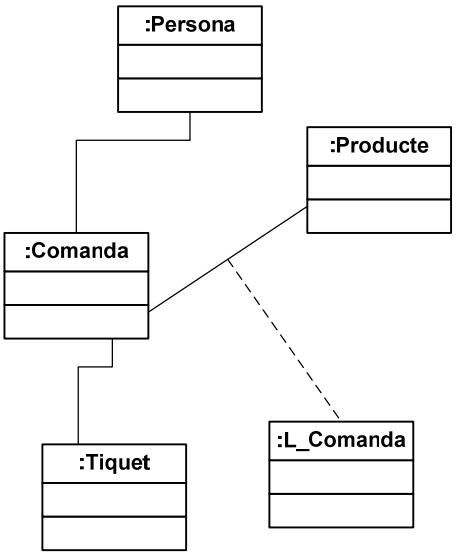
Començarem a parlar del **botiguer**: aquesta persona és l'encarregada d'atendre als clients, formalitzar les comandes d'aquests i realitzar les vendes.

Pel què fa al **pastisser**, la seva feina és la d'elaborar els productes sol·licitats pels clients, és a dir, un cop el botiguer ha formalitzat la comanda li passa la nota al pastisser que elaborarà els productes necessaris.

En últim lloc, parlaré del **propietari,** la seva feina és la de contractar el personal emmagatzemar la informació sobre els diferents proveïdors, establir els preus de les matèries primeres i comprovar els beneficis que s'han fet.

2.1.2. Model de domini

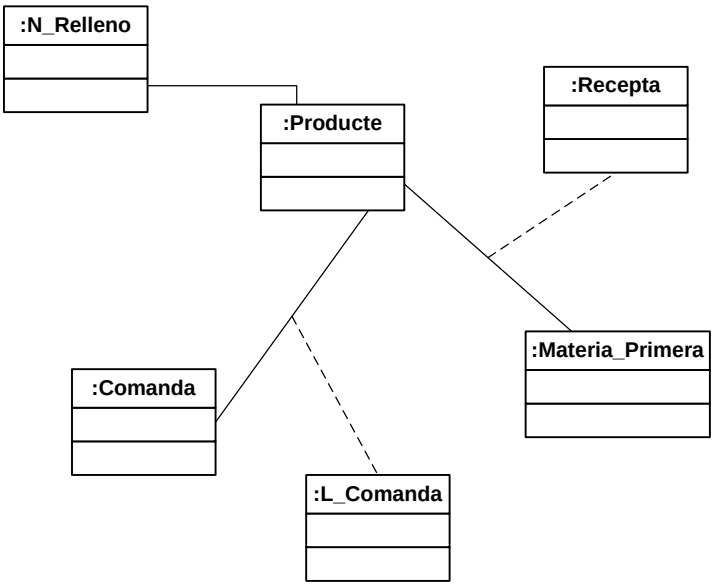
Seguidament, es pot observar un diagrama d'objectes que ens permet identificar quines són les classes més destacades de la pròpia aplicació.
Aquest model de domini és diferent depenent de qui utilitzi l'aplicació. En primer lloc, es mostrarà el model de domini del **botiguer**:



Model de Domini: diagrama d'objectes del botiguer

Una **persona** realitza una comanda que està formada per un o més productes. Un producte i una quantitat forma part del què s'anomena **L_comanda.** Quan una comanda està feta i la persona que l'ha sol·licitada la va a recollir, es fa un **tiquet** on hi figura el cost de tota la comanda.

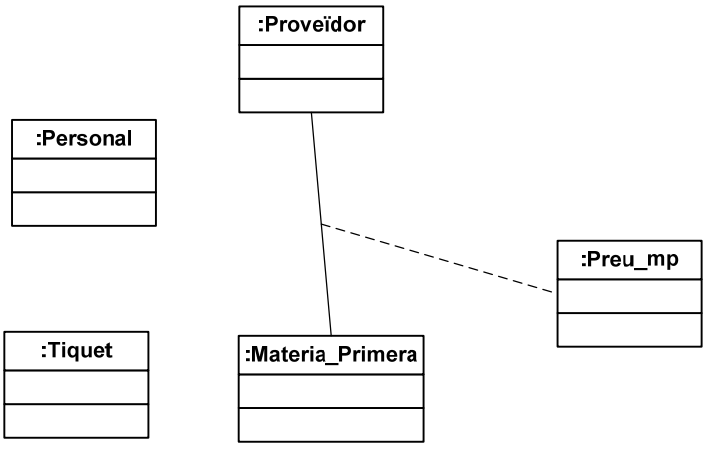
Un cop vist el model de domini del botiguer, mostraré el corresponent al **pastisser**:



Model de Domini: Diagrama d'objectes del pastisser

Un cop s'ha formalitzat la **comanda**, aquesta es passa al pastisser. A cada **L_Comanda** es pot veure el producte i la quantitat que s'ha sol·licitat. I, mitjançant la **recepta,** es pot calcular la quantitat de **matèria primera** necessària per fer la comanda.

Per acabar els models de domini, és hora de parlar del model que li correspon al **propietari**:



Model de Domini: Diagrama d'objectes del propietari

El propietari és el que s'ocupa de treballar amb **proveïdors,** cadascun d'ells pot tenir les mateixes o diferents **matèries primeres,** tot i que cada proveïdor té els seus propis preus. A més, s'ocupa de contractar **personal** i revisar els **tiquets.**

2.1.3. Model de negoci

Seguidament, es descriuran alt per alt les funcions que necessita l'empresa que ha d'utilitzar aquest *software*. Per mostrar-ho, s'utilitzarà el diagrama de casos d'ús:

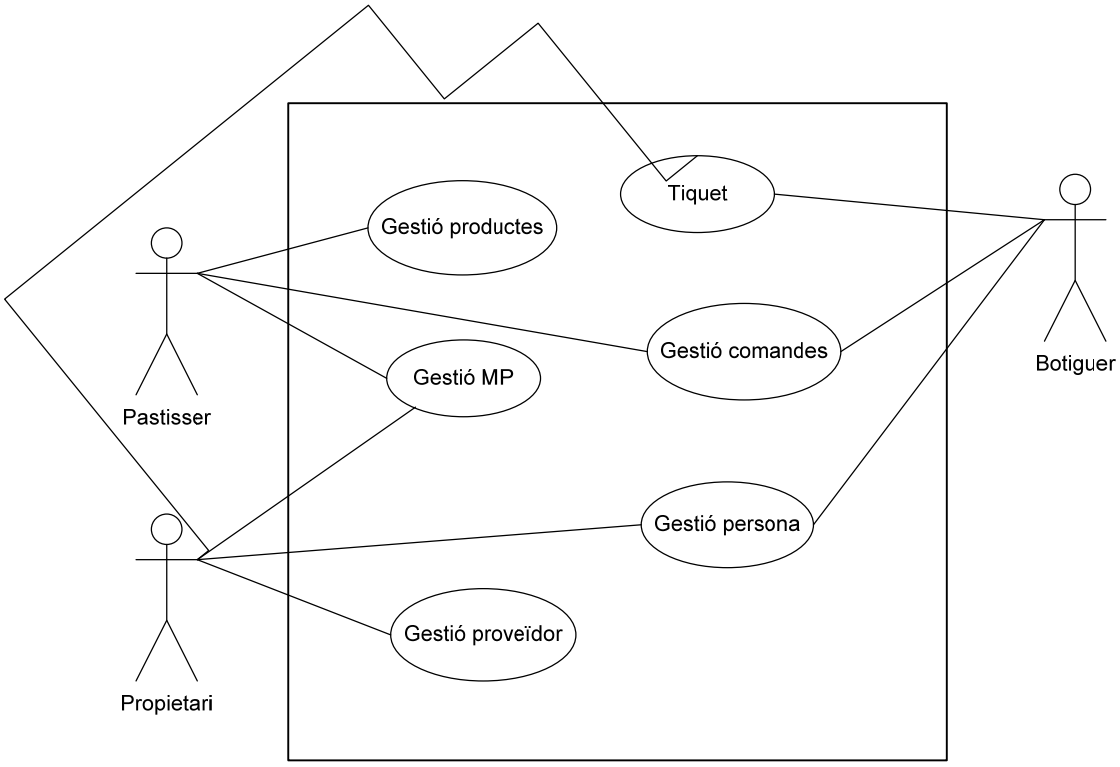


Diagrama de casos d'ús model de domini

En aquest diagrama, hi ha les diferents opcions que tindran els usuaris del programa. El pastisser té accés a la gestió de productes, de matèries primeres i de comandes. El propietari té accés a la gestió de matèries primeres, de proveïdors, de persones i dels tiquets. Per acabar el model de negoci, el botiguer pot accedir a la gestió de comandes, tiquets i persona.

Més endavant, concretaré les accions específiques que poden realitzar els tres tipus d'usuaris.

Un cop estudiats els requeriments de l'aplicació a desenvolupar, es procedirà a desenvolupar l'anàlisi del sistema.

3. Anàlisis del sistema

A continuació, es representarà de manera formal quines són les necessitats de l'organització. S'utilitzarà una metodologia orientada a objectes (UML).
Estarà compost per:

- **Model Estàtic**: apareixen els objectes persistents d'interès del domini. S'utilizarà un diagrama de classes i s'exemplificarà mitjançant diversos diagrames d'objectes.

- **Model Dinàmic**: s'especificarà la funcionalitat del sistema:
  - **SDEU(Sistema Discret d'Estat Únic):**
    - Diagrames de casos d'ús.
    - Diagrames de seqüències.

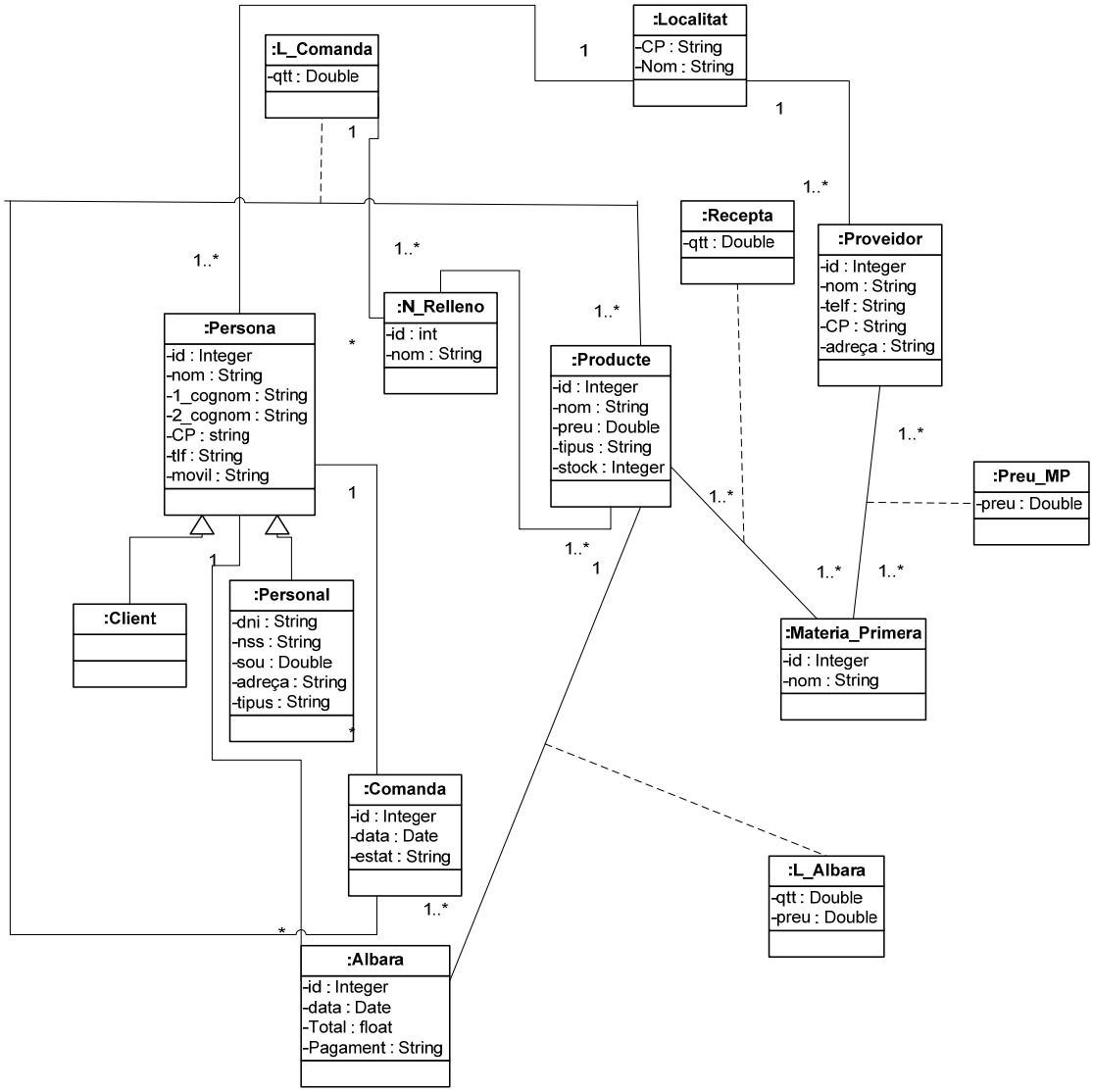
- Per acabar, s'ampliarà el **Diagrama de classes** del model estàtic mitjançant les propietats obtingudes en el desenvolupament del model dinàmic. I es farà un diagrama de classes corresponent a les finestres i l'herència que hi ha entre elles.

3.1. Model estàtic

Com s'ha comentat anteriorment, s'utilitzarà el **Diagrama de Classes** i s'exemplificarà amb diversos **Diagrama d'objectes**.

3.1.1. Diagrama de classes

Aquest diagrama permet representar tots els objectes d'interès del domini amb els seus atributs i també les relacions que hi ha entre ells. El diagrama obtingut és el següent:



Model Estàtic: diagrama de classes

3.1.2. Diagrama d'objectes

Seguidament, s'exemplificarà el diagrama anterior mitjançant un diagrama d'objectes.

Per què sigui més entenedor, es dividirà per parts, és a dir, es farà un diagrama sobre la recepta d'un producte i el seu farciment, sobre el preu d'una matèria primera i una sol·licitud d'una comanda.

- **Exemple1:** Recepta d'un producte i el seu farciment:

Hi ha un producte anomenat *prod7* que està format per la matèria primera *ous* (que té l'identificador 1) en una quantitat de *10 unitats* i la matèria primera llet (amb l'identificador *2)* amb una quantitat de *20 litres*.

Aquest producte pot estar farcit o no, en el cas de estar-ho, pot ser de *crema* o *nata* (amb els identificadors 1 i 2 respectivament). En el cas de no portar cap farciment, l'identificador és el 0. El preu del producte és de 1.25 € el quilo i és de fabricació pròpia.

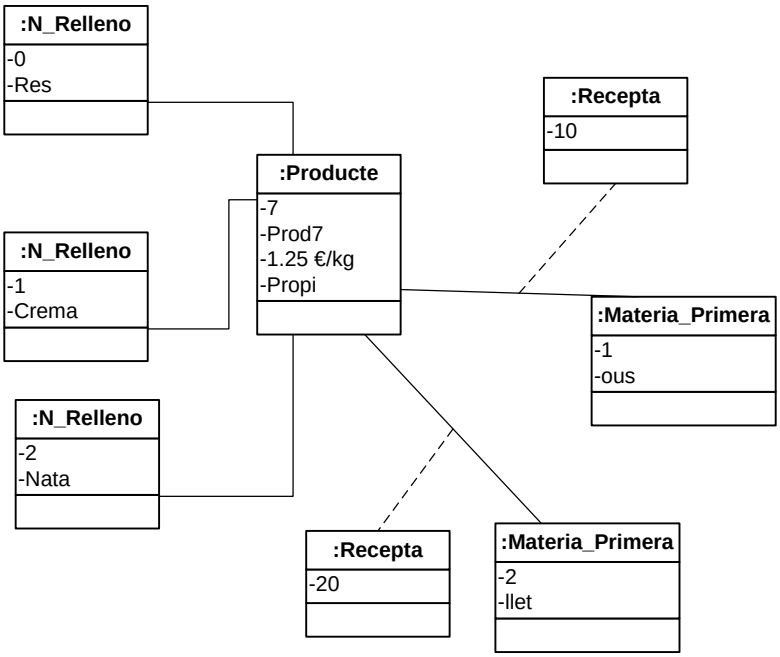


Diagrama d'objectes de la recepta d'un producte i el seu farciment

En aquest diagrama es pot observar la relació que hi ha entre els objectes. En l'exemple, el producte és el *prod7* amb les seves dades. Com hem comentat anteriorment, hi ha tres tipus de farciment que cadascun d'ells forma un objecte, i tots estan relacionats amb el producte.

En el cas de la recepta, cada producte té associat una o diverses matèries primeres en una determinada quantitat. Per tant, es necessita un objecte per cada matèria primera i, en la relació que hi ha amb el producte, se li posa una classe associativa que ens indica la quantitat.

- **Exemple 2:** Relació entre matèria primera, proveïdor i preu.

Hi ha dos proveïdors, el primer anomenat *prov1* (amb adreça c/sense nom de 08600 Berga i el telèfon 123456789) ens ofereix la matèria primera *sucre* (que val 12 € el sac) i la *llet* (que val 0.30€/l), amb els identificadors 3 i 2 successivament.
El segon proveïdor, *prov2*, (amb adreça c/major 08670 i el telèfon 987654321) ens torna a oferir la *llet* (a 0.35€/l) i, a més, ens ofereix *cacau* amb l'identificador 4 (que val 1.20€/kg).

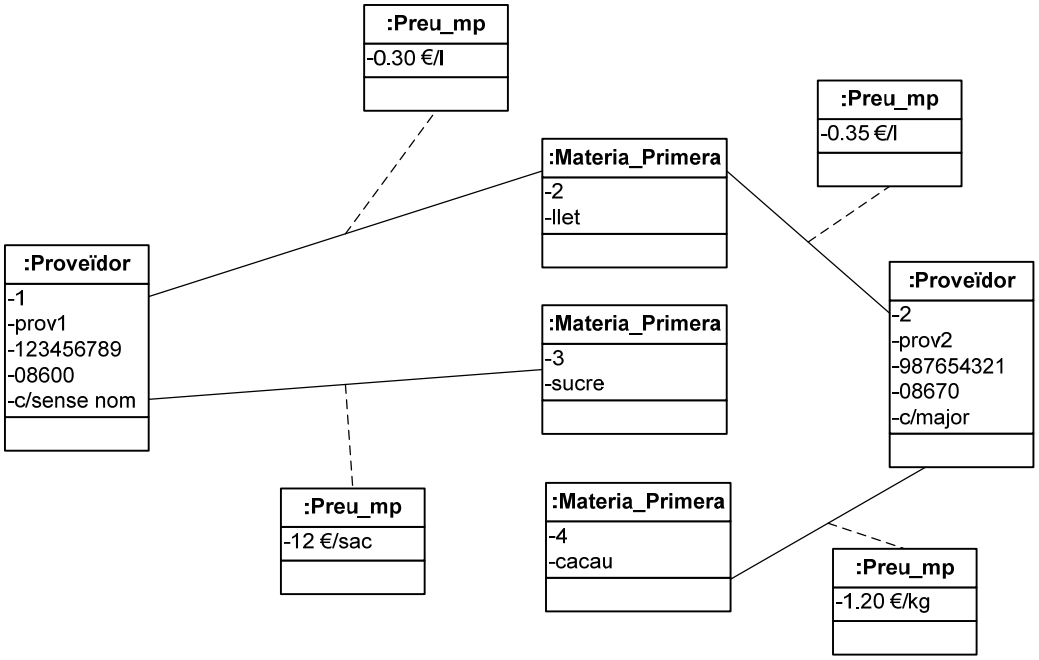


Diagrama d'objectes de la relació entre MP, proveïdor i preu

En aquest exemple, hi ha dos proveïdors que formem dos objectes i les matèries primeres en formen tres més. En la relació que hi ha entre les matèries primeres i els proveïdors, s'indica el preu mitjançant una classe associativa.

Exemple 3: Realització d'una comanda.

El client número 1, que es diu Joan Mateos Alonso (que viu al Pstg. Del plom 22 del Vendrell 43700 i amb el telèfon 123456789 i mòbil 666666666) ha realitzat una comanda que anirà a recollir el dia xx/xx/2009. En aquesta comanda ha demanat 2

Kg del *prod7* i 2.5 Kg de *prod8*. Un segon client ha realitzat una comanda de 3 Kg del *prod8* que recollirà el dia yy/xx/2009.

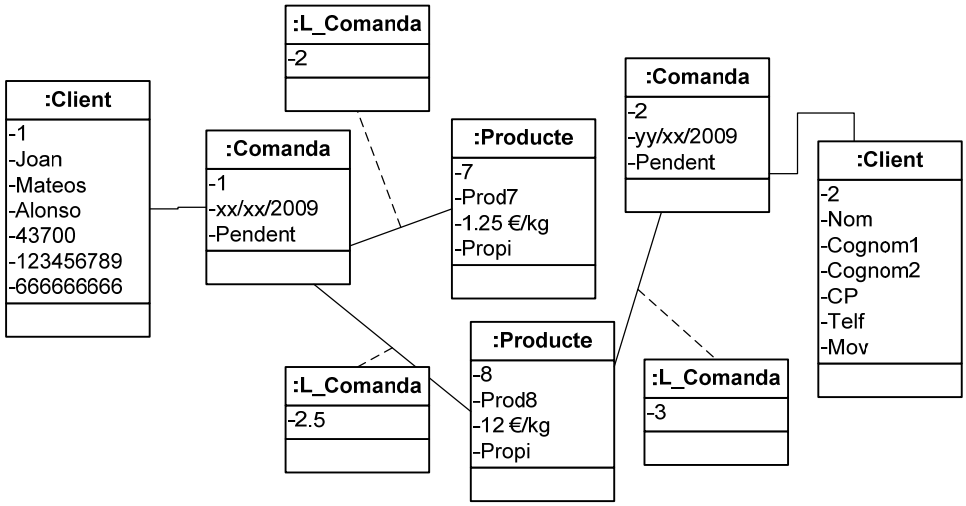


Diagrama d'objectes de realització d'un comanda

En aquest cas, l'objecte *comanda* equival a la capçalera de la comanda. La relació que hi ha entre els objectes *producte* i *comanda* formen les línies de la comanda on s'indica la quantitat necessària de producte. La relació que hi ha entre el *client* i la *comanda*, serveix per saber qui l'ha fet.

3.2. Model Dinàmic

En l'apartat anterior s'ha obtingut informació sobre els diferents elements que seran necessaris per realitzar l'aplicació, sense tenir en compte les accions que duran a terme. En aquest apartat, es parlarà amb més detall de les diferents accions que ha de realitzar el sistema.

En primer lloc, es definiram diversos **diagrames de casos d'ús** on es representaran les accions que ha de dur a terme el sistema. A més, com ja s'ha comentat en l'especificació dels requeriments, s'ha d'afegir el manteniment dels diferents objectes. Per què el resultat sigui més entenedor, es començarà amb un cas d'us molt compactat i s'anirà desglossant fins realizar-los a nivell d'esdeveniment

3.2.1 Diagrames de casos d'ús

En primer lloc, es dividiran totes les accions en tres grups, que seran els de **manteniment, terminal punt de venda (TPV) i calcular MP per comanda**, com mostra el primer diagrama de casos d'ús.

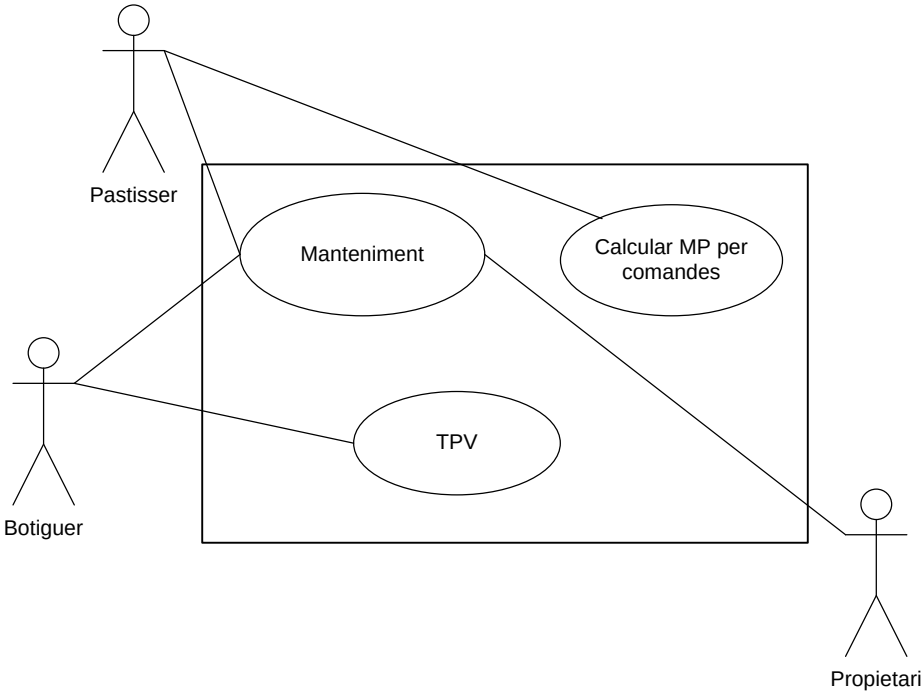


Diagrama de cas d'ús inicial

El cas de **manteniment**, hi poden accedir tots tres tipus d'usuaris (pastisser, botiguer i propietari), les **utilitats** són pensades pel pastisser i, per tant, només hi tindran accés ells. En últim lloc, al **TPV** hi pot accedir el botiguer.

A continuació, s'analitzarà el cas d'us de **manteniment** que és el més complex. Aquest cas es dividirà en nou casos d'ús corresponents al manteniment de producte, personal, client, comanda, matèria primera, proveïdor, localitat, recepta i farciment. En el diagrama següent es pot veure quin tipus d'usuari pot accedir a cada cas.

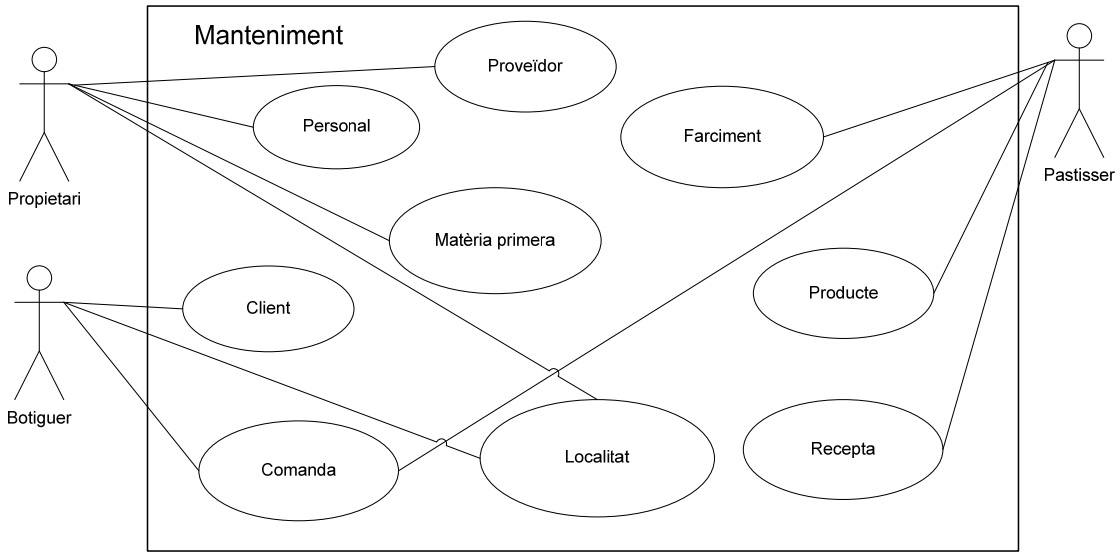


Diagrama de cas d'ús de manteniment

Un cop vistos tots els casos d'ús que hi ha dins de **manteniment** s'ha de desglossar cadascun d'ells i s'obtindran tots els casos d'ús a nivell d'esdeveniment.

- Cas d'ús de **personal**:

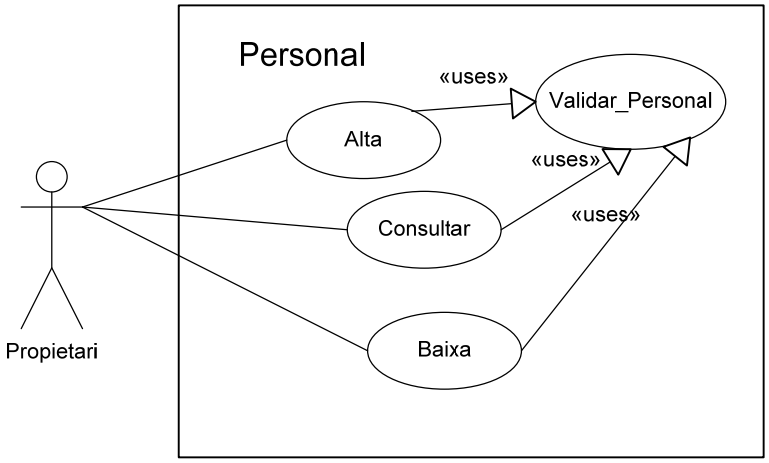


Diagrama de cas d'ús de manteniment de personal

En aquest cas només hi pot accedir el propietari, està dividit en tres casos d'ús que són els d'alta, baixa i consulta. Tots tres casos utilitzen l'acció de *validar_persona* per comprovar si existeix o no.

- Cas d'ús de **client**:

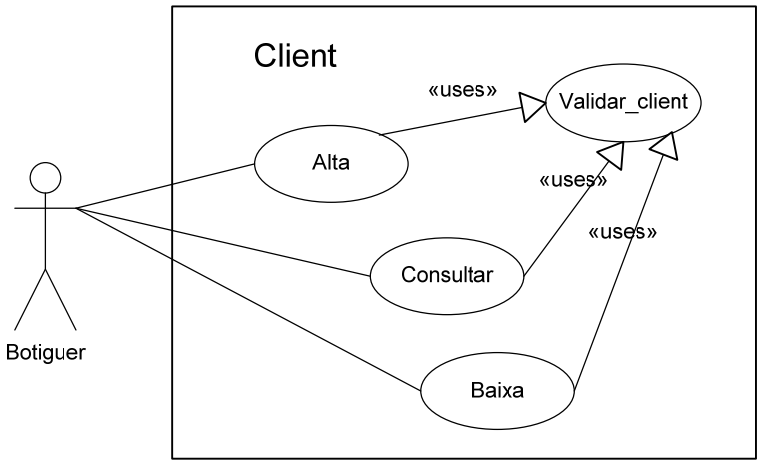


Diagrama de cas d'ús de manteniment de client

Aquest cas és molt similar a l'anterior, la diferència principal és que només pot donar d'alta el botiguer. La funció de *validar* és la mateixa que en el cas anterior.

- Cas d'ús del **producte**:

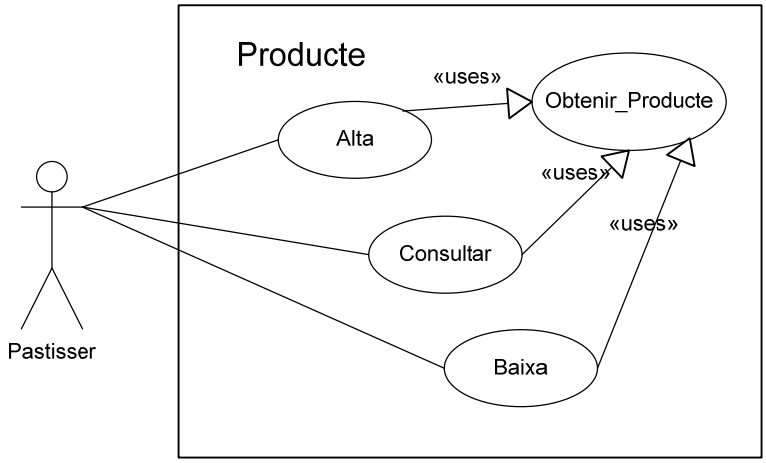


Diagrama de cas d'ús de manteniment de producte

Aquest cas d'ús l'utilitza el pastisser: té els permisos per donar d'alta, baixa i consultar els productes. Tots tres casos d'ús utilitzen una altra funció anomenada *obtenir_producte* que intenta obtenir el producte, si l'obté retorna *cert* i en el cas contrari retorna *fals*. Aquesta funció és l'equivalent a la *validar* de **personal** i **client**.

- Cas d'ús de **matèria primera (MP)**:

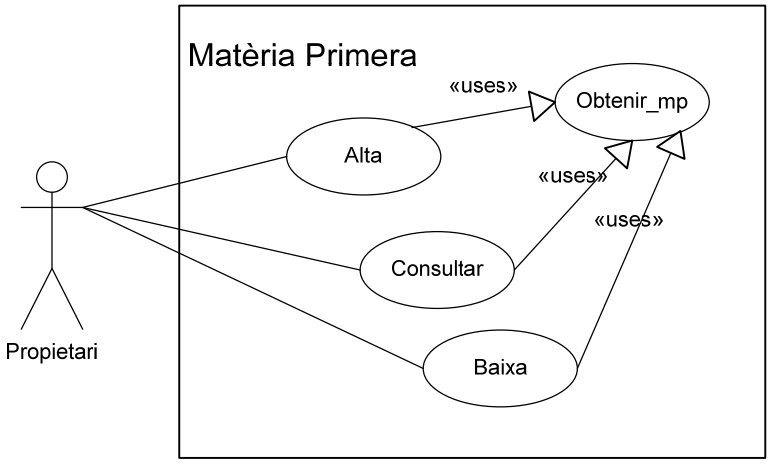


Diagrama cas d'ús de manteniment de matèria primera:

Aquest cas d'ús l'utilitza el propietari, té els permisos totals per treballar amb l'objecte **matèria primera**. Com passava amb el producte, la funció *obtenir_mp* serveix per comprovar si existeix o no i també segueix el mateix procediment.

- Cas d'ús de **recepta**.

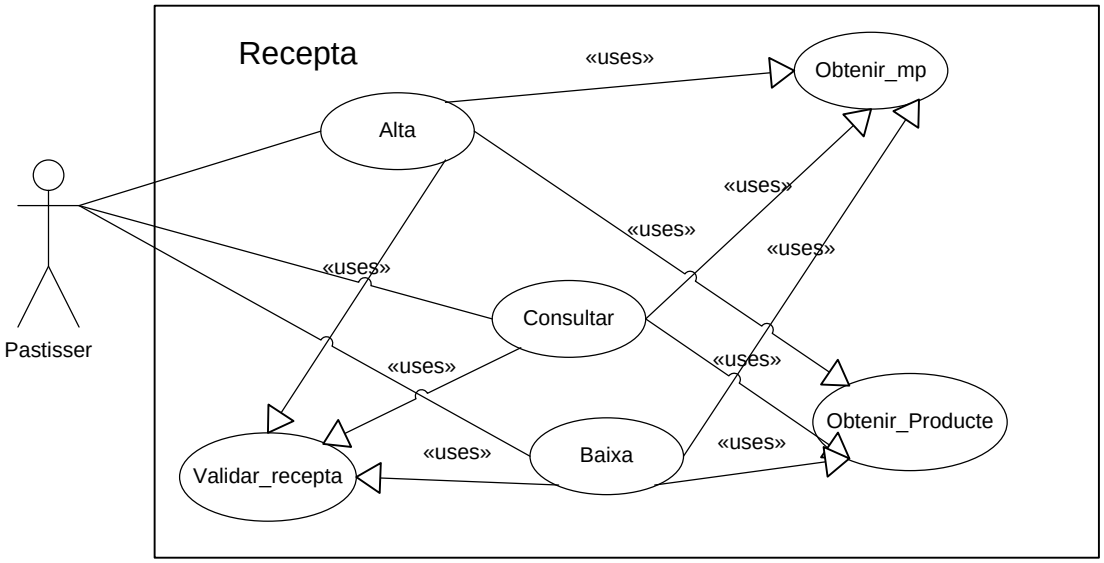


Diagrama de cas d'ús de manteniment de recepta

Aquest objecte el fa servir el pastisser. Com ja s'ha vist en el **diagrama de classes en el model estàtic**, la classe recepta està relacionada amb les classes

producte i matèria primera. En totes aquestes relacions es necessitarà validar-ho mitjançant la funció *obtenir_producte* i *obtenir_mp* que ja hem comentat en els diagrames anteriors. A més, s'utilitza el *validar_recepta* per comprovar que no existeixi aquesta entrada.

- Cas d'ús de **localitat**.

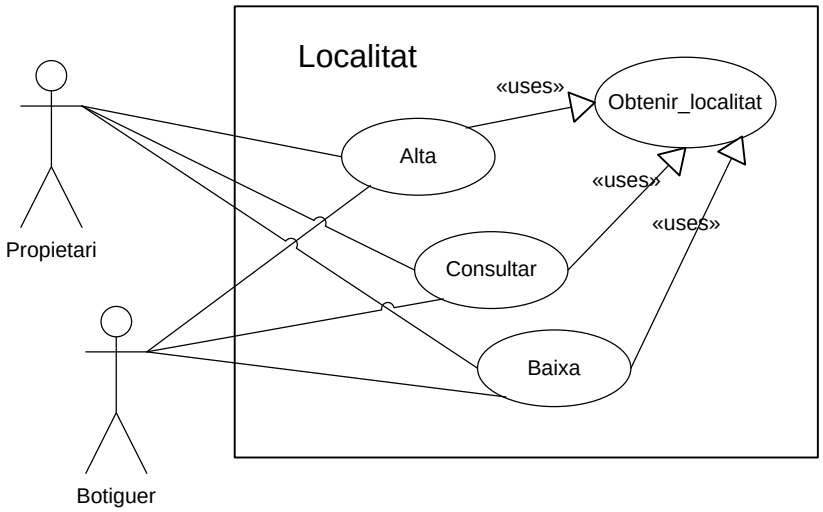


Diagrama cas d'ús de manteniment de localitat

En aquest objecte hi pot treballar tant el botiguer com el propietari, ja que són els encarregats de treballar amb els objectes personal, client i proveïdor. També podem observar que tots els esdeveniments utilitzen *obtenir_localitat*, la seva funció serà la de validar si la localitat existeix o no.

- Cas d'ús **proveïdor:**

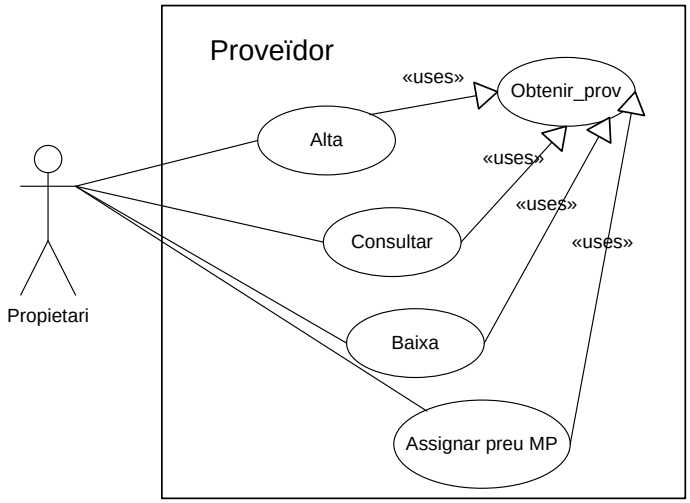


Diagrama cas d'ús de manteniment de proveïdor

Aquest esdeveniment l'utilitza el propietari. A més de tenir els esdeveniments d'alta, de baixa i de consulta, també incorpora un altre esdeveniment: *Assignar preu MP*. La funció d'aquest és, donat un proveïdor, assignar el preu d'una matèria primera. Tots aquest esdeveniments utilitzen la funció *obtenir_prov* com en els casos anteriors, serveix per comprovar si aquell proveïdor existeix o no.

- Cas d'ús **comanda**:

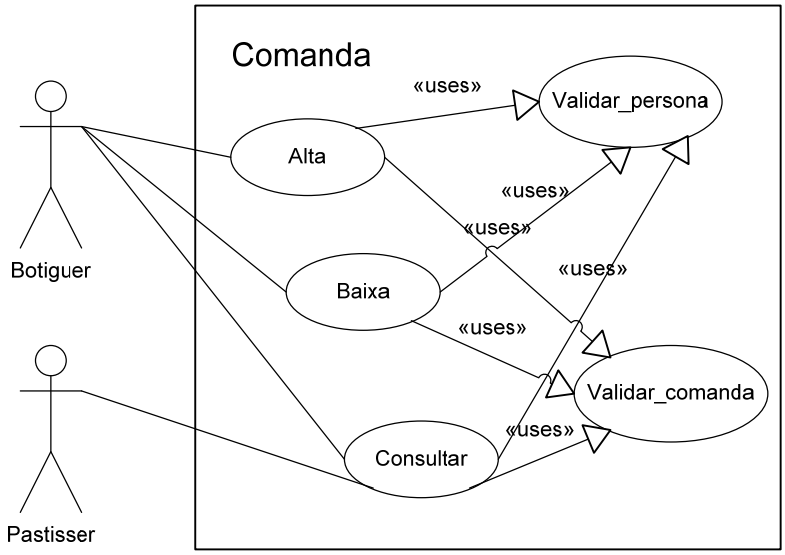


Diagrama de cas d'ús de manteniment de comanda

En aquest cas d'ús hi tenen accés dos usuaris: el botiguer i el pastisser, però tenen diferents privilegis. El botiguer pot accedir a qualsevol esdeveniment i, en canvi, el pastisser només pot accedir a la consulta. Aquests casos d'ús utilitzen dos esdeveniments que són el de *validar_persona* (serveix per comprovar si la persona existeix o no) i el de *validar_comanda* (per comprovar que no hi hagi cap identificador de la comanda repetit).

- Cas d'ús **Farciment** (en el diagrama de classes *N_Relleno*):

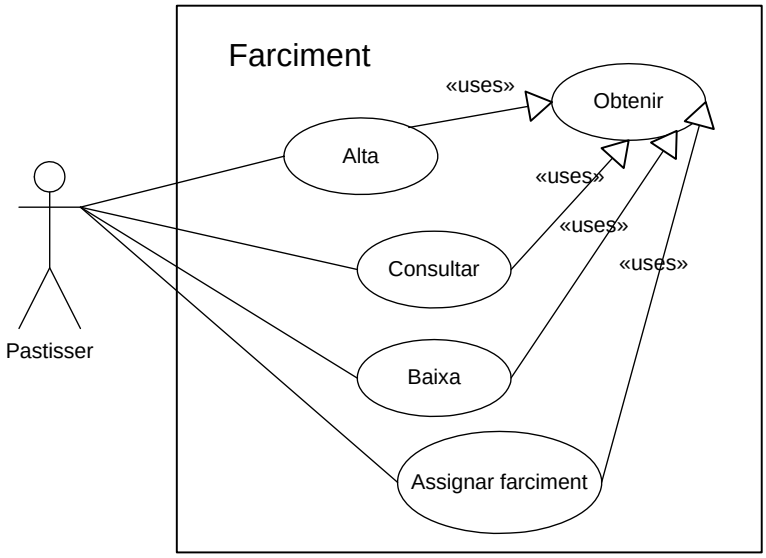


Diagrama de cas d'ús de manteniment de farciment

En aquest cas d'ús només hi pot accedir el pastisser, ja que és l'encarregat de farcir els diferents productes. A més de l'alta, la baixa i la consulta també hi ha un altre esdeveniment que és el d'*assignar farciment*. La seva funció és dir quins farcits poden tenir els diferents productes. Tots aquest casos utilitzen la funció *obtenir*, que ens indica si el farciment existeix o no.

Si tornem al primer diagrama, anomenat **diagrama cas d'ús inicial**, veiem que encara hi ha dos casos d'ús més. Tan el **TPV** com **calcular MP per comanda** són a nivell d'esdeveniment.

A continuació, es descriuran el diferents casos d'ús.

- Cas d'ús: TPV.
  - **Resum de la funcionalitat:** realitzar vendes d'un o diversos productes.
  - **Casos d'ús relacionats:** cap.
  - **Precondició:** un botiguer serveix la petició d'un client.
  - **Postcondició:** realitzar la venda dels productes.
  - **Descripció del flux d'entrada:** productes i quantitat.

- Cas d'ús: Calcular MP per comandes.
    - **Resum Funcionalitat:** calcular la quantitat de matèries primeres necessàries per satisfer un període de comandes.
    - **Casos d´ús relacionats:** cap.
    - **Precondició:** el pastisser vol saber quines matèries primeres necessita per fer les comandes.
    - **Postcondició:** llistat on es mostra el nom i la quantitat de matèries primeres necessaris per satisfer les comandes.
    - **Descripció de flux d'entrada:** dues dates que ens indiquen el període de temps en que es vol calcular.

- Cas d'ús: Alta personal.
    - **Resum de funcionalitat:** donar d'alta el personal.
    - **Casos d'ús relacionats:** *Validar_personal*.
    - **Precondició:** el propietari vol contractar personal per l'empresa.
    - **Postcondició:** registrar el nou personal.
    - **Descripció del flux d'entrada:** tota la informació necessària per donar d'alta la persona.

- Cas d'ús: Baixa personal.
    - **Resum de funcionalitat:** esborrar el personal.
    - **Casos d'ús relacionats:** *Validar_personal*.
    - **Precondició:** el propietari vol donar de baixa al personal.
    - **Postcondició:** esborrar el personal de la base de dades.
    - **Descripció del flux d'entrada:** el propietari consulta el personal i el dóna de baixa.

- Cas d'ús: Consultar personal.
    - **Resum de funcionalitat:** obtenir informació sobre el personal.
    - **Casos d'ús relacionats:** *Validar_personal.*
    - **Precondició:** el propietari en vol obtenir informació.
    - **Postcondició:** informació sobre ell.
    - **Descripció del flux d'entrada:** nom, cognoms o tipus personal.

- Cas d'ús: Validar personal.
    - **Resum de funcionalitat:** comprovar si existeix o no.
    - **Casos d'ús relacionats:** cap**.**
    - **Precondició:** donar d'alta, de baixa o consultar.
    - **Postcondició:** si existeix o no.
    - **Descripció del flux d'entrada:** nom i cognoms.

- Cas d'ús: Alta client.
    - **Resum de funcionalitat:** donar d'alta un client.
    - **Casos d'ús relacionats:** *Validar_client*.

- **Precondició:** el botiguer vol donar d'alta un client.
- **Postcondició:** registrar el nou client.
- **Descripció del flux d'entrada:** tota la informació necessària per donar d'alta la persona.

- Cas d'ús: Baixa client.
  - **Resum de funcionalitat:** esborrar un client.
  - **Casos d'ús relacionats:** *Validar_client*.
  - **Precondició:** el botiguer vol donar de baixa el client.
  - **Postcondició:** esborrar el client de la base de dades.
  - **Descripció del flux d'entrada:** el botiguer consulta el client i el dóna de baixa.

- Cas d'ús: Consular client.
  - **Resum de funcionalitat:** obtenir informació sobre el client.
  - **Casos d'ús relacionats:** *Validar_client.*
  - **Precondició:** el botiguer vol obtenir informació.
  - **Postcondició:** informació sobre el client.
  - **Descripció del flux d'entrada:** nom, cognoms.

- Cas d'ús: Validar client.
  - **Resum de funcionalitat:** comprovar si existeix o no.
  - **Casos d'ús relacionats:** cap.
  - **Precondició:** donar d'alta, de baixa o consultar el client.
  - **Postcondició:** si existeix o no.
  - **Descripció del flux d'entrada:** nom i cognoms.

- Cas d'ús: Alta producte.
  - **Resum de funcionalitat:** donar d'alta un producte.
  - **Casos d'ús relacionats:** *obtenir_prod*.
  - **Precondició:** el pastisser dóna d'alta un producte.
  - **Postcondició:** registrar el nou producte.
  - **Descripció del flux d'entrada:** tota la informació necessària per donar d'alta el producte.

- Cas d´ús: Baixa producte.
  - **Resum de funcionalitat:** esborrar un producte.
  - **Casos d'ús relacionats:** *Obtenir_producte*.
  - **Precondició:** el pastisser vol donar de baixa el producte.
  - **Postcondició:** esborrar el producte de la base de dades.
  - **Descripció del flux d'entrada:** el pastisser consulta el producte i el dóna de baixa.

- Cas d'ús: Consultar producte.
  - **Resum de funcionalitat:** obtenir informació sobre el producte.
  - **Casos d'ús relacionats:** *Obtenir_prod*.
  - **Precondició:** el pastisser en vol obtenir informació.
  - **Postcondició:** informació sobre ell.
  - **Descripció del flux d'entrada:** identificador o nom.

- Cas d'ús: *Obtenir_prod*.
  - **Resum de funcionalitat:** comprovar si existeix o no. Si existeix omple l'objecte.
  - **Casos d'ús relacionats:** cap**.**
  - **Precondició:** donar d'alta, de baixa o consultar.
  - **Postcondició:** si existeix o no. En el cas que si, retornar-lo.
  - **Descripció del flux d'entrada:** identificador, nom.

- Cas d'ús: Alta matèria primera.
  - **Resum de funcionalitat:** donar d'alta la matèria primera .
  - **Casos d'ús relacionats:** *Obtenir_mp*.
  - **Precondició:** el propietari vol donar d'alta una matèria primera.
  - **Postcondició:** registrar la nova matèria primera.
  - **Descripció del flux d'entrada:** tota la informació necessària per donar d'alta la matèria primera.

- Cas d'ús: Baixa matèria primera.
  - **Resum de funcionalitat:** esborrar una matèria primera.
  - **Casos d'ús relacionats:** *Obtenir_mp*.
  - **Precondició:** el propietari vol donar de baixa la matèria primera.
  - **Postcondició:** esborrar la matèria primera de la base de dades.
  - **Descripció del flux d'entrada:** el propietari la consulta i la dóna de baixa.

- Cas d'ús: Consultar matèria primera.
  - **Resum de funcionalitat:** obtenir informació sobre la matèria primera.
  - **Casos d'ús relacionats:** *Obtenir_mp*.
  - **Precondició:** el propietari en vol obtenir informació.
  - **Postcondició:** informació sobre la matèria primera.
  - **Descripció del flux d'entrada:** identificador o nom.

- Cas d'ús: *Obtenir_mp.*
  - **Resum de funcionalitat:** comprovar si existeix o no. Si existeix l'obté.
  - **Casos d'ús relacionats:** cap.
  - **Precondició:** donar d'alta, de baixa o consultar.

- **Postcondició:** si existeix o no.
- **Descripció del flux d'entrada:** identificador, nom.

- Cas d'ús: Alta recepta.
  - **Resum de funcionalitat:** el pastisser vol donar d'alta la constitució d'un producte.
  - **Casos d'ús relacionats:** *Obtenir_prod, Obtenir_mp, validar_recepta.*
  - **Precondició:** el pastisser vol afegir informació sobre la constitució del producte.
  - **Postcondició:** registrar la nova recepta.
  - **Descripció del flux d'entrada:** tota la informació necessària per donar d'alta la recepta.

- Cas d'ús: Baixa recepta.
  - **Resum de funcionalitat:** esborrar un recepta.
  - **Casos d'ús relacionats:** *Obtenir_prod, Obtenir_mp, validar_recepta.*
  - **Precondició:** el pastisser vol donar de baixa la línia de la recepta.
  - **Postcondició:** esborrar la línia de la base de dades.
  - **Descripció del flux d'entrada:** el pastisser consulta la recepta del producte i l'esborra.

- Cas d'ús: Consultar recepta.
  - **Resum de funcionalitat:** obtenir informació sobre la recepta d'un producte.
  - **Casos d'ús relacionats:** *Obtenir_prod, Obtenir_mp, validar_recepta.*
  - **Precondició:** el pastisser en vol obtenir informació.
  - **Postcondició:** informació sobre ell.
  - **Descripció del flux d'entrada:** identificador o nom del producte.

- Cas d'ús: Validar recepta.
  - **Resum de funcionalitat:** comprovar si existeix o no la relació entre el producte i la matèria primera.
  - **Casos d'ús relacionats:** cap**.**
  - **Precondició:** donar d'alta, de baixa o consultar.
  - **Postcondició:** si existeix o no.
  - **Descripció del flux d'entrada:** identificador del producte i identificador de la matèria primera.

- Cas d'ús: Alta localitat.
  - **Resum de funcionalitat:** el botiguer o el propietari vol donar d'alta una localitat.

- **Casos d'ús relacionats:** *Obtenir_localitat*.
- **Precondició:** el botiguer o el propietari dona d'alta la localitat.
- **Postcondició:** registrar la nova localitat.
- **Descripció del flux d'entrada:** tota la informació necessària per donar d'alta la localitat.

- Cas d'ús: Baixa localitat.
  - **Resum de funcionalitat:** esborrar una localitat.
  - **Casos d'ús relacionats:** *Obtenir_localitat*.
  - **Precondició:** el propietari o el botiguer vol esborrar la localitat.
  - **Postcondició:** esborrar la localitat de la base de dades.
  - **Descripció del flux d'entrada:** el propietari o el botiguer consulta la localitat i l'esborra.

- Cas d'ús: Consultar localitat.
  - **Resum de funcionalitat:** obtenir informació sobre la localitat.
  - **Casos d'ús relacionats:** *Obtenir_localitat*.
  - **Precondició:** el botiguer o el propietari volen obtenir informació sobre la localitat.
  - **Postcondició:** informació sobre ella.
  - **Descripció del flux d'entrada:** codi postal o nom de la localitat.

- Cas d'ús: Obtenir localitat.
  - **Resum de funcionalitat:** comprovar si existeix o no la localitat. Si existeix l'obté.
  - **Casos d'ús relacionats:** cap.
  - **Precondició:** donar d'alta, de baixa o consultar.
  - **Postcondició:** si existeix o no.
  - **Descripció del flux d'entrada:** codi postal o nom de la localitat.

- Cas d'ús: Alta proveïdor.
  - **Resum de funcionalitat:** el propietari vol donar d'alta una un proveïdor.
  - **Casos d'ús relacionats:** *Obtenir_prov*.
  - **Precondició:** el propietari dona d'alta un proveïdor.
  - **Postcondició:** registrar el nou proveïdor.
  - **Descripció del flux d'entrada:** tota la informació necessària per donar d'alta el proveïdor.

- Cas d'ús: Baixa proveïdor.
  - **Resum de funcionalitat:** esborrar un proveïdor.
  - **Casos d'ús relacionats:** *Obtenir_prov*.
  - **Precondició:** el propietari vol esborrar un proveïdor.
  - **Postcondició:** esborrar el proveïdor de la base de dades.

- **Descripció del flux d'entrada:** el propietari consulta el proveïdor i l'esborra.

- Cas d'ús: Consultar proveïdor.
  - **Resum de funcionalitat:** obtenir informació sobre el proveïdor.
  - **Casos d'ús relacionats:** *obtenir_prov.*
  - **Precondició:** el propietari volen obtenir informació sobre la el proveïdor.
  - **Postcondició:** informació sobre ell.
  - **Descripció del flux d'entrada:** nom o identificador.

- Cas d'ús: *Obtenir_prov*.
  - **Resum de funcionalitat:** comprovar si existeix o no el proveïdor. Si existeix l'obté.
  - **Casos d'ús relacionats:** cap**.**
  - **Precondició:** donar d'alta, de baixa o consultar.
  - **Postcondició:** si existeix o no.
  - **Descripció del flux d'entrada:** nom o identificador.

- Cas d'ús: Alta comanda.
  - **Resum de funcionalitat:** Un client ha sol·licitat uns productes i el botiguer fa la comanda.
  - **Casos d'ús relacionats:** *Validar_comanda* i *Validar_persona*.
  - **Precondició:** el botiguer dóna d'alta una comanda.
  - **Postcondició:** registrar la nova comanda.
  - **Descripció del flux d'entrada:** informació sobre el client, sobre el dia de recollida i els productes sol·licitats.

- Cas d'ús: Baixa comanda.
  - **Resum de funcionalitat:** esborrar una comanda.
  - **Casos d'ús relacionats:** *Validar_comanda* i *Validar_persona*.
  - **Precondició:** un client no vol la comanda, sempre i quant no estigui ja feta.
  - **Postcondició:** esborrar la comanda de la base de dades.
  - **Descripció del flux d'entrada:** el botiguer consulta la comanda i la dóna de baixa sempre i quan no estigui feta.

- Cas d'ús: Consultar comanda.
  - **Resum de funcionalitat:** obtenir informació sobre la comanda, tan el botiguer com el pastisser.
  - **Casos d'ús relacionats:** *Validar_comanda* i *Validar_persona*.
  - **Precondició:** el botiguer o el client volen veure la informació sobre la comanda.

- **Postcondició:** informació de la comanda juntament amb la informació del client i les línies de comanda.
- **Descripció del flux d'entrada:** identificador de la comanda o un interval de dates o nom o cognoms del client.

- Cas d'ús: *Validar_comanda*.
  - **Resum de funcionalitat:** comprovar si l'identificador existeix o no.
  - **Casos d'ús relacionats:** cap.
  - **Precondició:** donar d'alta, de baixa o consultar.
  - **Postcondició:** si existeix o no.
  - **Descripció del flux d'entrada:** identificador.

- Cas d'ús: *Validar_persona*.
  - **Resum de funcionalitat:** comprovar si existeix una persona o no.
  - **Casos d'ús relacionats:** cap.
  - **Precondició:** donar d'alta, de baixa o consultar.
  - **Postcondició:** si existeix o no.
  - **Descripció del flux d'entrada:** nom o cognoms.

- Cas d'ús: Alta farciment.
  - **Resum de funcionalitat:** el pastisser dóna d'alta un farciment pels productes.
  - **Casos d'ús relacionats:** *Obtenir*.
  - **Precondició:** el pastisser vol donar d'alta un nou farciment.
  - **Postcondició:** registrar el nou farciment.
  - **Descripció del flux d'entrada:** el nom del farciment nou.

- Cas d'ús: Baixa farciment.
  - **Resum de funcionalitat:** esborrar un farciment.
  - **Casos d'ús relacionats:** *Obtenir*.
  - **Precondició:** el pastisser vol donar de baixa el farciment i no hi ha cap producte que l'utilitzi.
  - **Postcondició:** esborrar el farciment de la base de dades.
  - **Descripció del flux d'entrada:** el pastisser el consulta i el dóna de baixa.

- Cas d'ús: Consultar farciment.
  - **Resum de funcionalitat:** obtenir informació sobre el farciment
  - **Casos d'ús relacionats:** *Obtenir*.
  - **Precondició:** el pastisser en vol obtenir informació.
  - **Postcondició:** informació sobre el farciment.
  - **Descripció del flux d'entrada:** identificador o nom.

- Cas d´ús: *Obtenir*.
    - **Resum de funcionalitat:** comprovar si existeix o no el farciment. Si existeix l'obté.
    - **Casos d'ús relacionats:** cap.
    - **Precondició:** donar d'alta, de baixa o consultar.
    - **Postcondició:** si existeix o no.
    - **Descripció del flux d'entrada:** identificador, nom

3.2.2 Diagrames de seqüències

Seguint amb el **SDEU**(Sistema Discret d'Estat Únic), un cop descomposats els diferents casos d'ús fins a nivell d'esdeveniment, es detallarà d'una manera acurada la funció de cada cas a nivell d'esdeveniment. S'utilitzarà el **diagrama de seqüència** que ens permet veure les accions que s'han de dur a terme per tal d'aconseguir els resultats desitjats. Es procurarà que hi hagi una encapsulació per què els diferents objectes siguin el màxim d'independents, propietat necessària per l'orientació a objectes. Per tant, es crearan unes classes frontera que són la pròpia interfície gràfica del sistema.

Com que tan el **personal** com els **clients** tenen els mateixos esdeveniments (l'única cosa que els diferencia són els atributs), el següents diagrames es podran generalitzar pels dos. Es parlarà de manteniment de **persona**, que equivaldrà a **personal** o **client**. Si es tracta de **client**, l'actor serà el botiguer i si es tracta de **personal**, serà el propietari.

- Diagrama de seqüència del **manteniment de la persona**

Es començarà per l'**alta**:

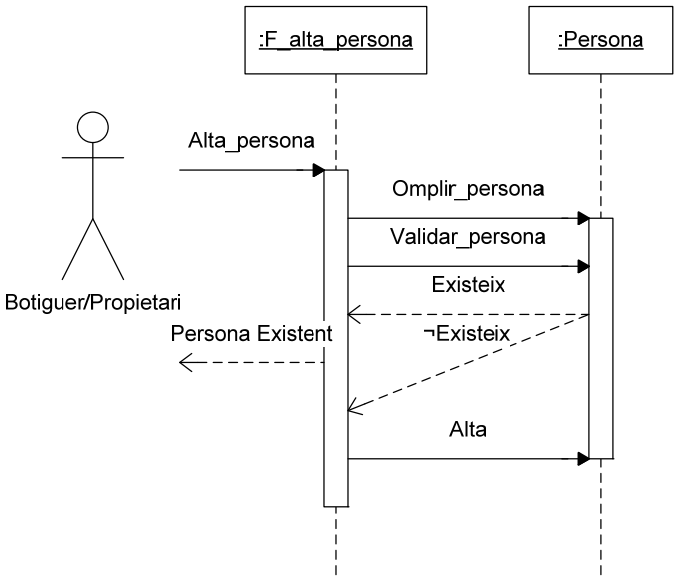


Diagrama seqüència de l'alta de la persona

En aquest esdeveniment hi intervé una classe frontera i un objecte. Quan es vol donar d'alta una persona, en primer lloc s'omple el propi objecte i es prossegueix a validar la persona. En el cas que no existeixi, es procedirà a donar-la d'alta. En el cas que no es pugui validar, es mostrarà per pantalla un missatge informant.

El següent diagrama de seqüències que s'analitzarà serà el de **Consultar persona:**

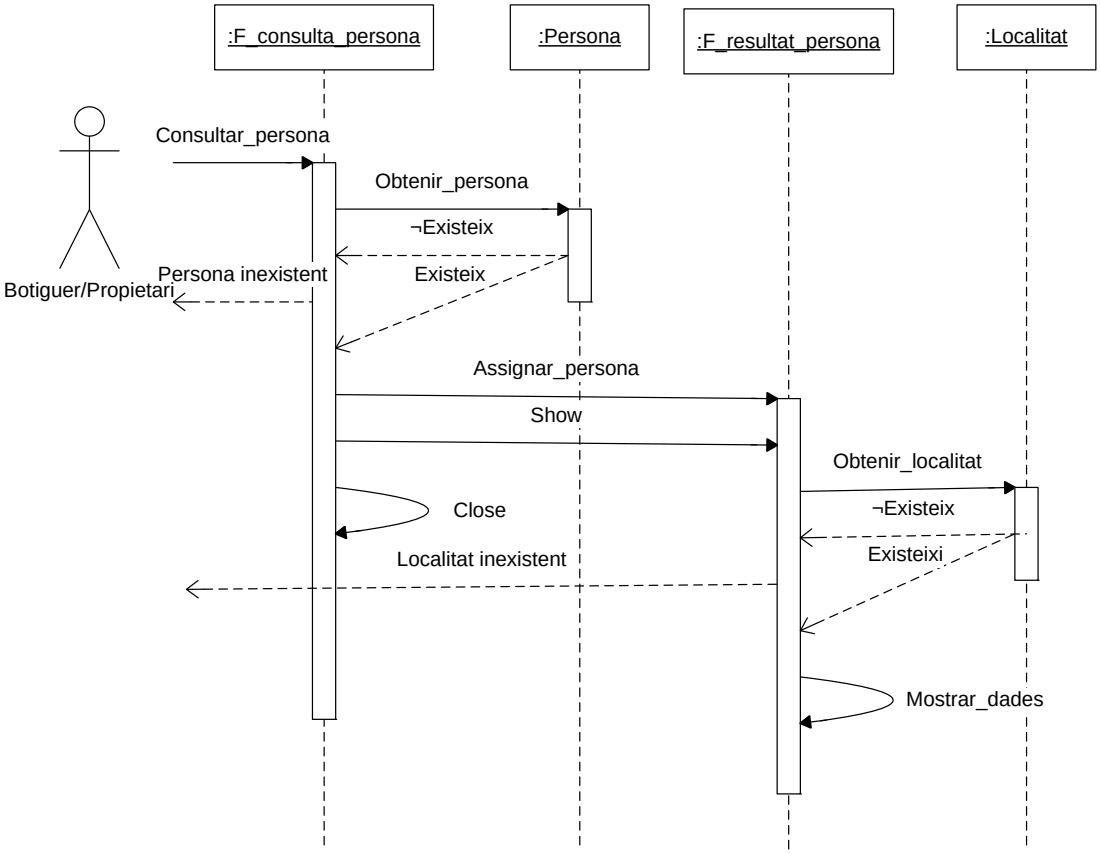
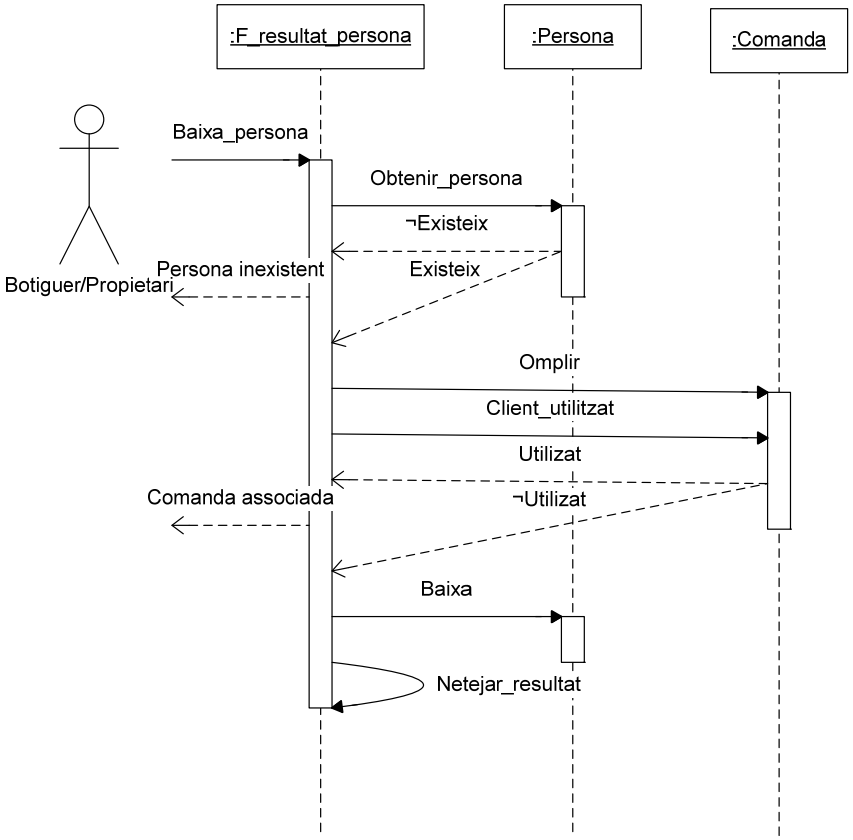


Diagrama de seqüència de la consulta de persona

Hi intervenen dues classes frontera i dos objectes. En primer lloc, s'intenta obtenir la persona. Un cop obtinguda, s'assigna a la següent classe frontera *F_resultat_persona* i es mostra la finestra. Un cop mostrada, s'intenta obtenir la localitat on pertany la persona. Si s'obté, es procedeix a mostrar la informació per pantalla. En el cas que hi hagi algun problema, es mostrà un missatge informant.

El següent esdeveniment serà el de **baixa persona:**



Digrama de seqüència baixa de persona

En aquest esdeveniment hi intervé una classe frontera i dos objectes. En primer lloc, s'intenta obtenir la persona. Un cop obtinguda, es procedeix a omplir una comanda com a màscara on només interessa l'identificador de la persona i el tipus. Un cop omplert, es comprova si aquella persona té alguna comanda associada, en el cas de no tenir-ne es donarà de baixa. En el cas que en tingui alguna, no es podrà donar de baixa. Si no es pot obtenir la persona, es mostrarà per pantalla un missatge informant del problema.

L'últim esdeveniment a analitzar en aquest cas d'ús és el de **modificar persona**:

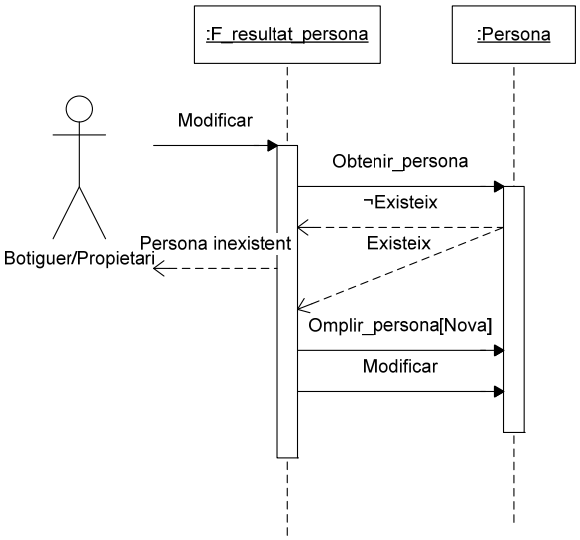


Diagrama de seqüència modificar persona

En aquest esdeveniment només hi intervé una classe frontera i un objecte. En primer lloc, s'intenta obtenir la persona a modificar. Un cop obtinguda, es procedeix a omplir una nova persona amb les noves dades. Un cop omplert es procedeix a la funció *modificar* que s'ha de passar com a paràmetre la nova persona.

- Diagrama de seqüència de **manteniment de producte**

Es començarà per l'**alta.**

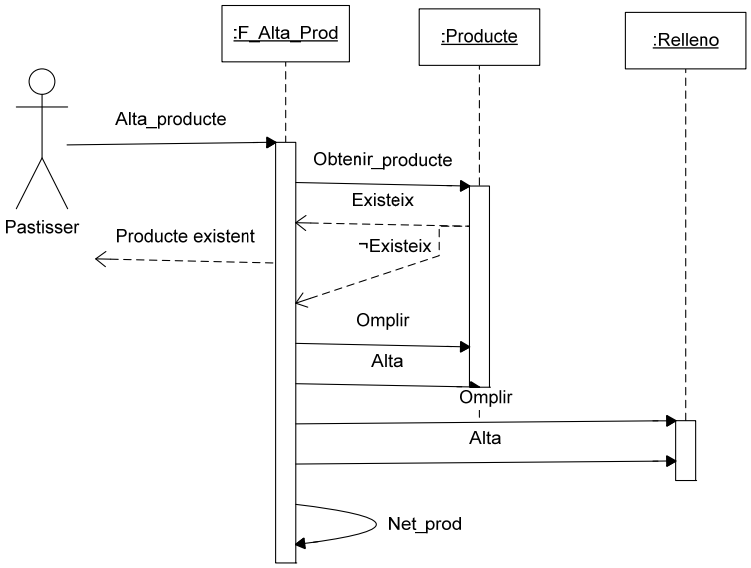


Diagrama seqüència d'alta de producte

En aquest esdeveniment hi intervenen tres tipus d'objectes, en primer lloc l'objecte *F_alta_prod* serà la classe frontera, és a dir, la interfície. El *producte* és la classe més important i, en últim lloc, hi ha l'objecte *relleno*, que només servirà en aquest cas per assignar el farciment. Per defecte, la base de dades porta l'estàndard *res.*

El passos a seguir seran els següents: quan el client prem el botó *guardar* de la finestra **alta producte**, s'intenta obtenir el producte amb el nom. Si no el pot obtenir, llavors omple l'objecte *producte* i el dóna d'alta a la base de dades. Un cop fet, es dóna d'alta també la relació entre aquest producte i el *farciment 0* (equival al farciment buit). Un cop realitzades totes aquestes operacions, es netejarà la pantalla. En cas d'obtenir el producte, es mostrarà un missatge a la pantalla que aquell producte ja existeix.

Seguidament es pralarà de l'esdeveniment **consultar producte.**

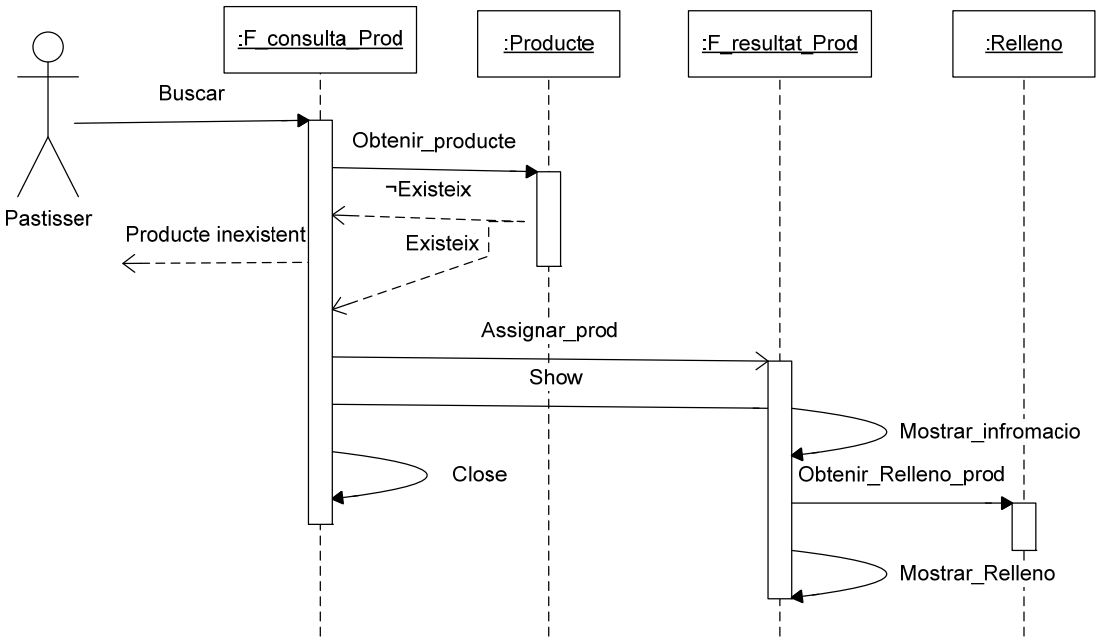


Diagrama de seqüència consultar producte

En aquest esdeveniment hi intervenen quatre objectes, dues classes frontera i dos objectes. En primer lloc, quan l'usuari busca la informació dóna l'ordre a la finestra *F_consulta_prod***.** Llavors s'intenta obtenir el producte, si s'obté, el copiarà a la finestra *F_resultat_prod* i la mostrarà. Aquesta serà l'encarregada de mostrar tota

la informació relacionada amb el producte, a més, s'obtindrà una llista on hi figuraran tots el farciments que estan relacionats amb aquest producte i també els mostrarà.

A continuació, toca parlar de l'esdeveniment **Baixa producte:**

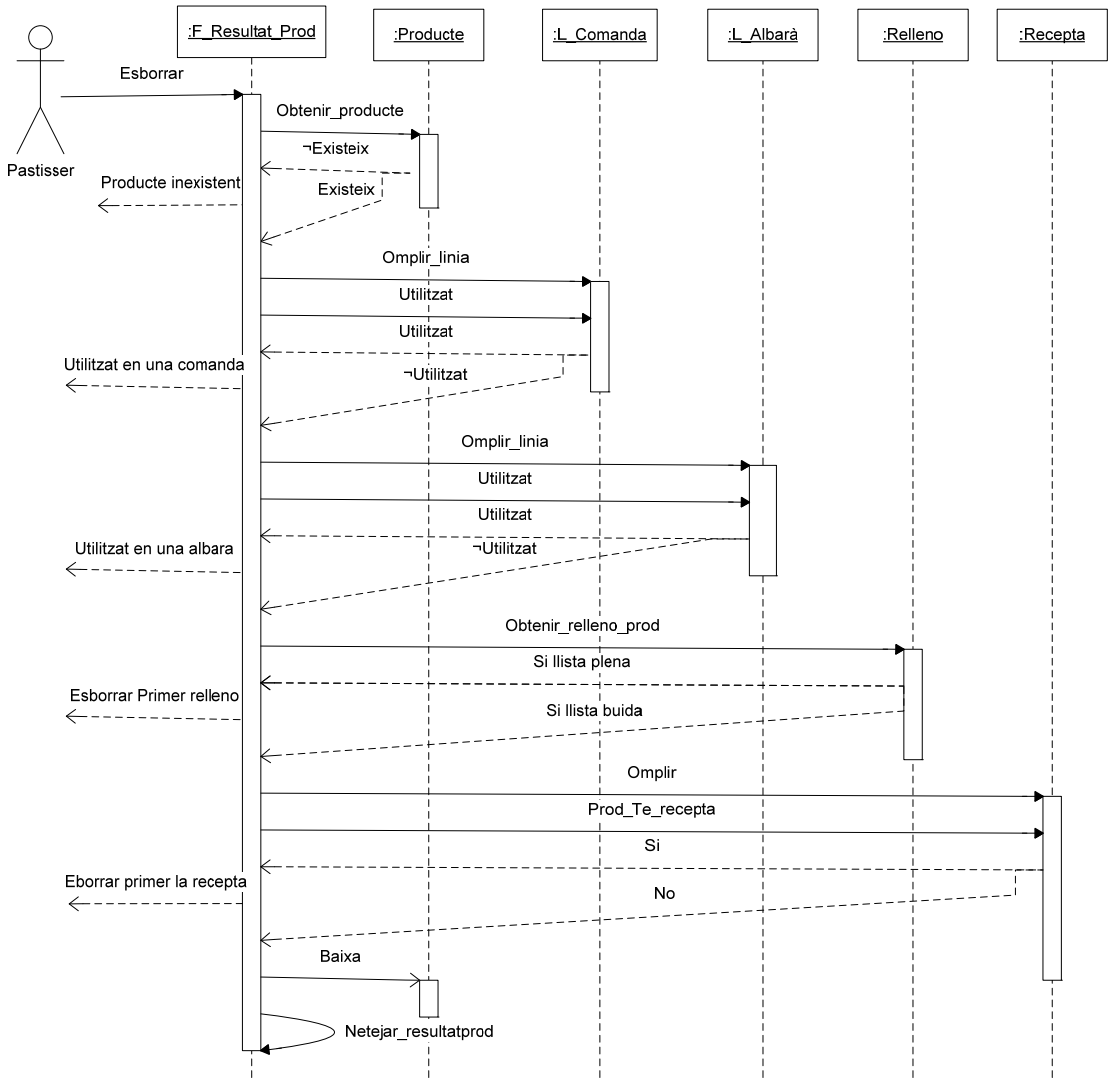


Diagrama de seqüència baixa producte

En aquest esdeveniment, s'utilitzen cinc objectes i una classe frontera. En primer lloc s'obté el producte. En el cas negatiu, es mostra el missatge *Producte inexistent*, en el cas contrari, procedeix a comprovar si aquell producte està en alguna comanda. En el cas que hi figuri, no es pot esborrar el producte. En el cas contrari, comprova si està en algun albarà i realitza el mateixos passos que en el cas anterior. En el últim lloc, es comprova si té algun farciment associat. En el cas

negatiu, es comprova si té alguna recepta introduïda. Si tampoc en té cap es procedeix a esborrar el producte de la base de dades.

L'últim esdeveniment és el de **modificar el producte:**

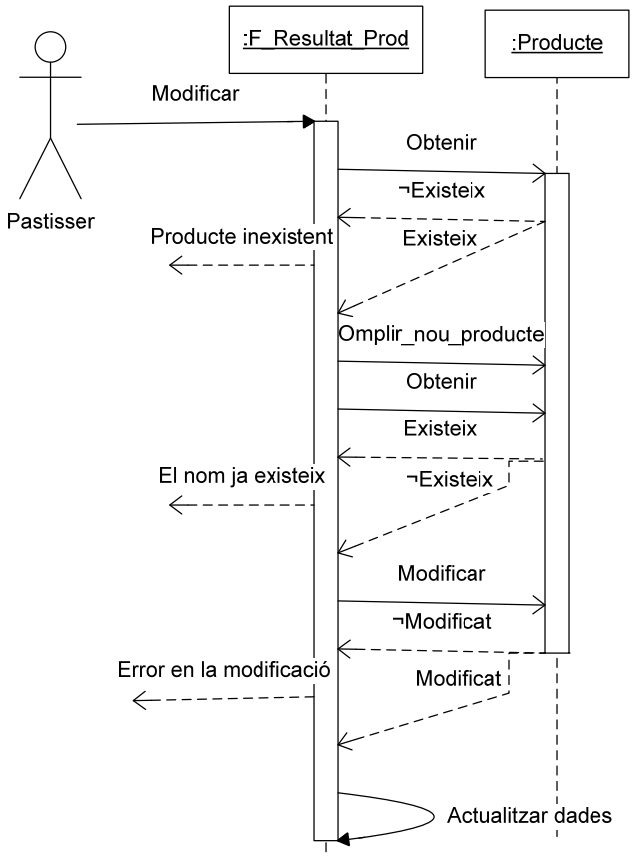


Diagrama de seqüència modificar producte

En aquest esdeveniment només hi intervé una classe frontera i un objecte. Quan l'usuari vol modificar la informació del producte, la finestra intenta obtenir-lo. En cas afirmatiu, omple un nou objecte amb les noves dades, comprova que aquest no existeixi (amb el mateix nom) i, en últim lloc, utilitza la funció *modificar* que necessita que li passis l'objecte nou. Un cop modificat, s'actualitzen les dades.

- Diagrama de seqüència del **manteniment de matèries primeres**

Es començarà **alta:**

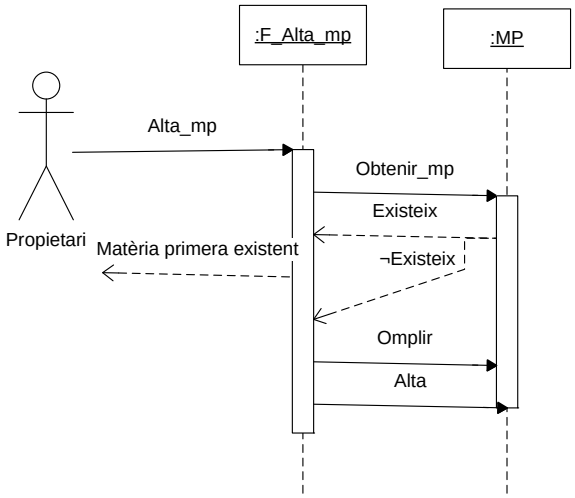


Diagrama de seqüència alta MP

Hi intervé una classe frontera i un objecte. En primer lloc, quan l'usuari dóna d'alta una matèria primera, s'intenta obtenir. Si no s'aconsegueix, es procedeix a omplir l'objecte i, en últim lloc, es dóna d'alta. En el cas contrari, es mostra un missatge informant que aquella matèria ja existeix.

A continuació, es mostrarà el diagrama corresponent a la consulta d'una **matèria primera**

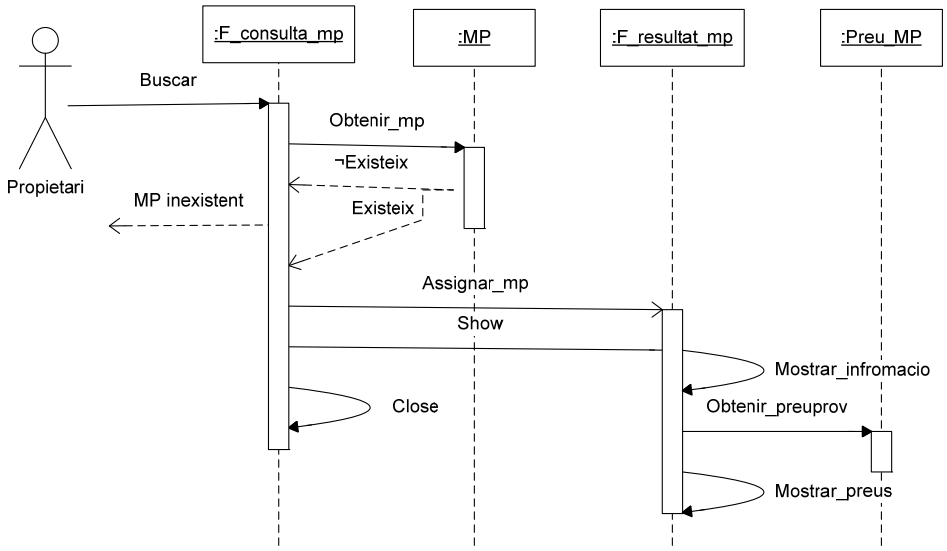


Diagrama de seqüència consultar MP

Hi intervenen dues classes fronteres i dos objectes. En primer lloc, quan l'usuari busca una matèria primera, el programa l'intenta obtenir. En el cas negatiu, mostra un missatge informant que no existeix. En el cas afirmatiu, assigna l'objecte *MP* a la següent classe frontera que mostra la informació i obté la llista de preus dels diferents proveïdors que ofereixen la matèria i la mostra.

Es procedirà a l'esdeveniment de **baixa de la matèria primera:**

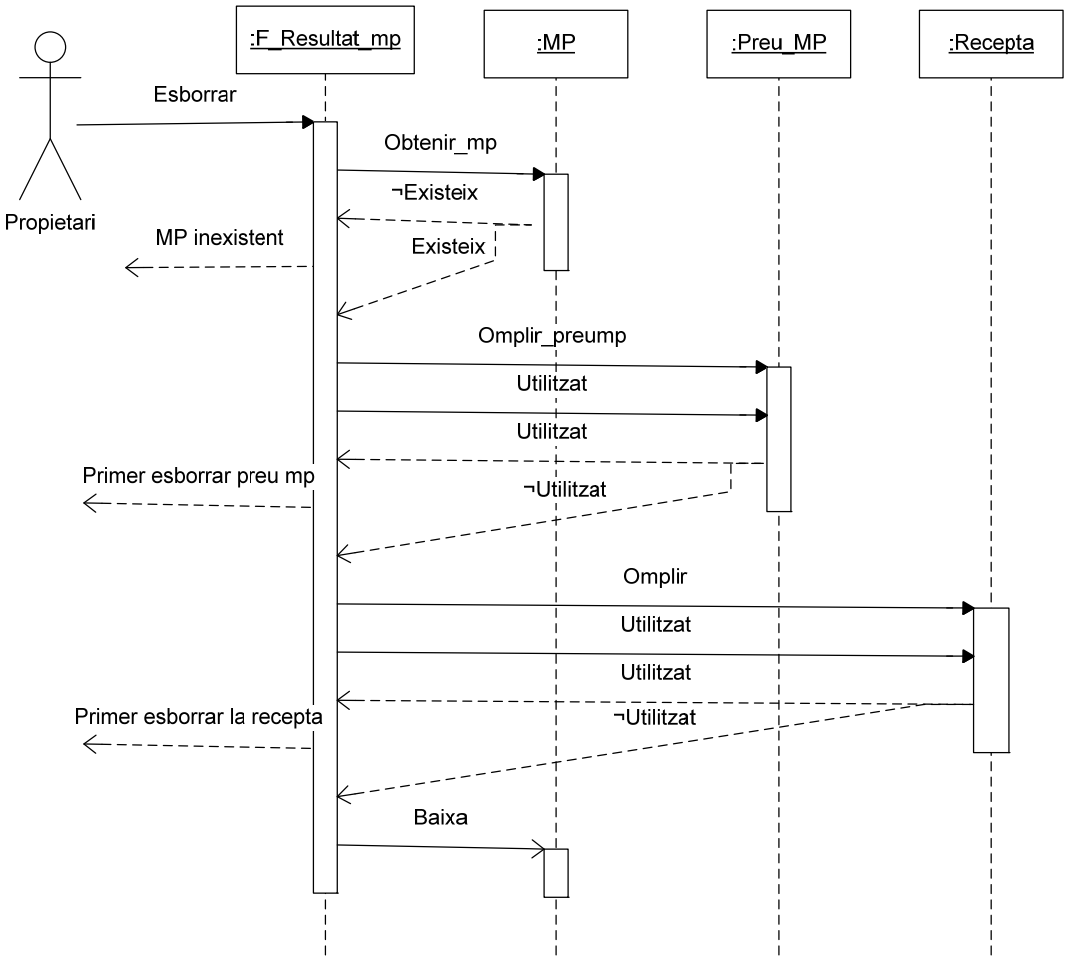


Diagrama seqüència baixa MP

Hi intervé una classe frontera i tres objectes. En el moment que l'usuari vol donar de baixa una matèria, s'intenta obtenir-la. En el cas afirmatiu, es comprova si té algun preu associat, si és així, es mostrarà per la pantalla un missatge informant que s'han d'esborrar les entrades dels preus. Sinó, es realitzarà la mateixa operació comprovant si té una recepta associada. Si no en té cap, es donarà de baixa la matèria. Sinó s'informarà que s'ha d'esborrar primer la recepta.

Per acabar aquest cas d'ús, ens falta l'esdeveniment **modificar matèria primera**:

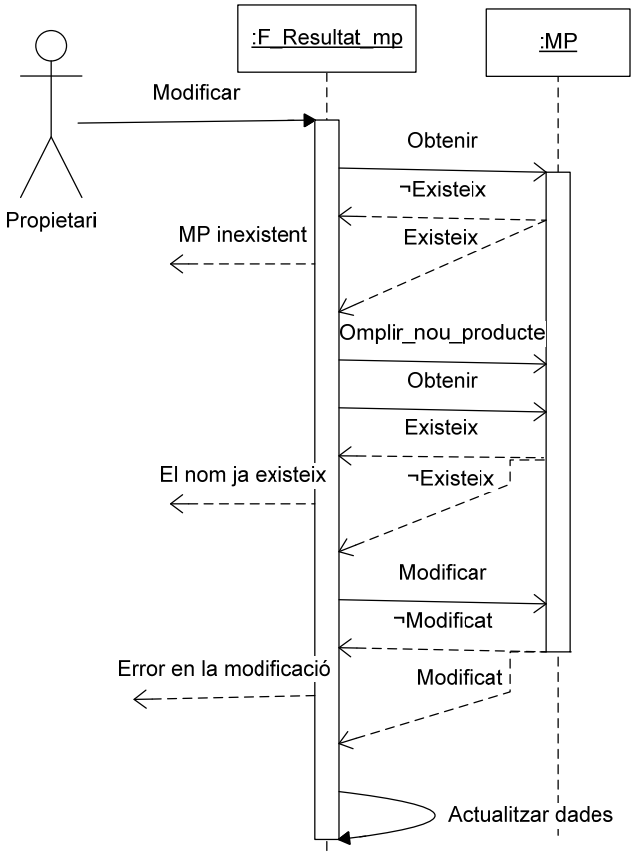


Diagrama seqüència modificar MP

Hi intervé una classe frontera i un objecte. Quan es vol modificar la matèria primera, en primer lloc el sistema l'intenta obtenir. En el cas afirmatiu, omple un nou objecte *MP* amb les dades noves, comprova que no existeix (amb el mateix nom) i utilitza la funció *modificar* que necessita el nou objecte creat. Es modifiquen les dades i s'actualitza la finestra.

- Diagrama de seqüència **manteniment de recepta**

Es començarà per l'**alta:**

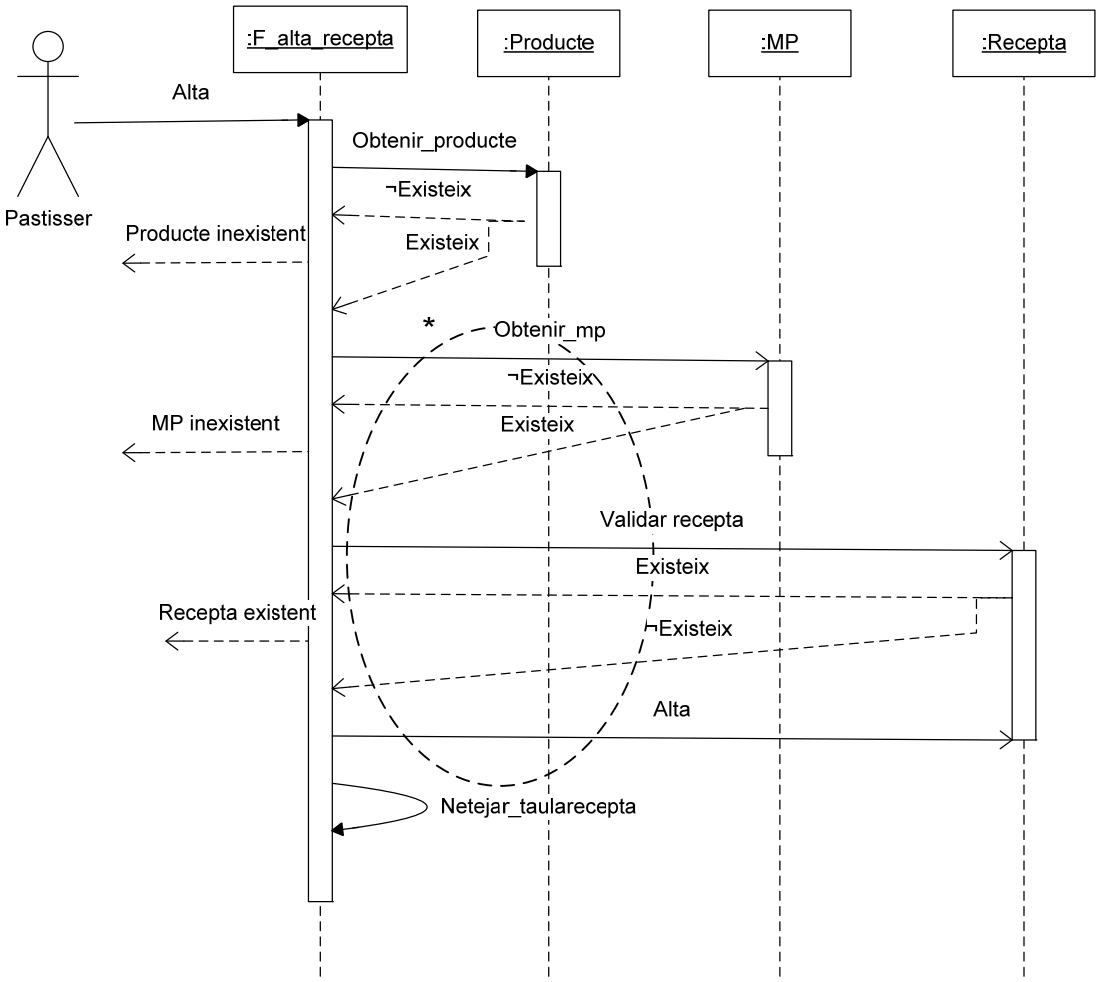


Diagrama seqüència alta Recepta

En aquest cas hi intervé una classe frontera i tres objectes. Quan es vol donar d'alta la recepta, en primer lloc s'intentarà obtenir el producte. En el cas positiu, començarà un bucle fins que s'acabi la llista. En aquest bucle es començarà intentant obtenir la matèria primera i, si s'ha pogut obtenir, es validarà si la recepta existeix o no. En el cas que no existeixi, es dóna d'alta, sinó ens informarà que aquella relació ja existeix.

A continuació, es mostrarà el diagrama corresponent a l'esdeveniment **consulta recepta**.

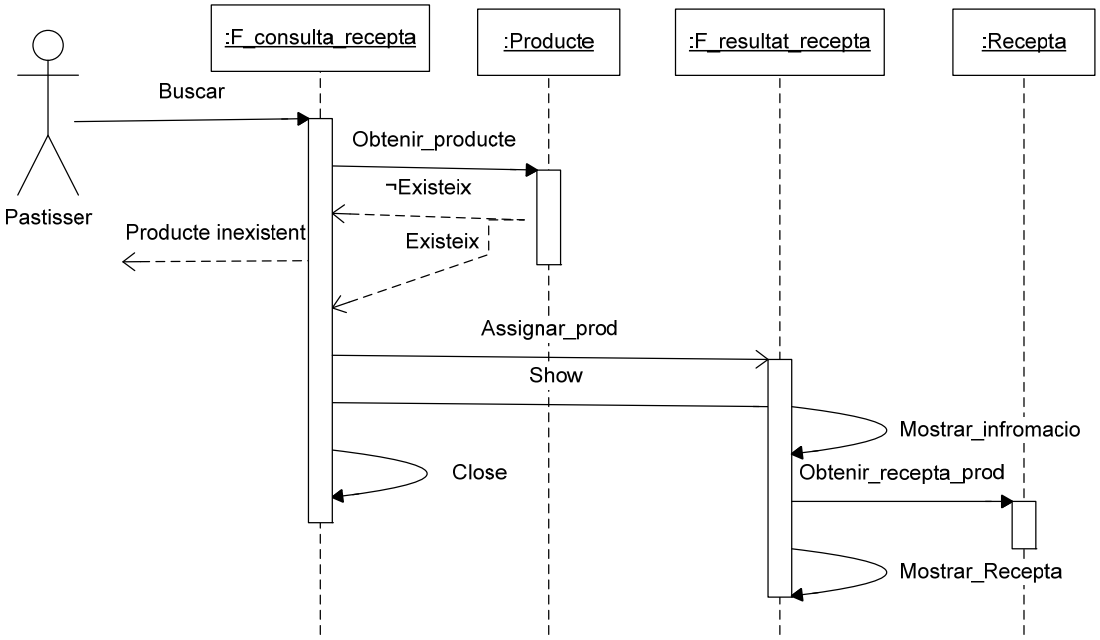


Diagrama seqüència consultar recepta

Hi intervenen dues classes frontera i dos objectes. Quan es vol consultar una recepta, en primer lloc s'intenta obtenir el producte. Si aquest s'obté, s'assigna a la següent classe frontera. Aquesta s'ocupa d'obtenir la recepta corresponent a aquell producte i mostrar-la.

Una utilitat que hem comentat al primer apartat és la de calcular la quantitat de matèries primers necessàries per fer una quantitat determinada de producte. Aquesta utilitat es troba a la mateixa pantalla de consulta, a la finestra del resultat hi ha una pestanya que permet fer els càlculs. A continuació, es mostrarà el diagrama corresponent a aquesta utilitat:

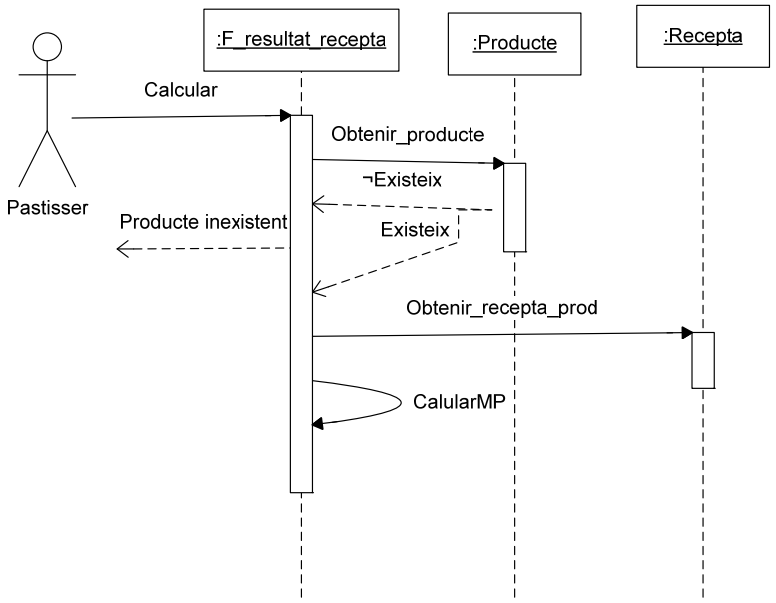


Diagrama seqüència utilitat

El següent esdeveniment a tractar serà la **baixa de recepta:**

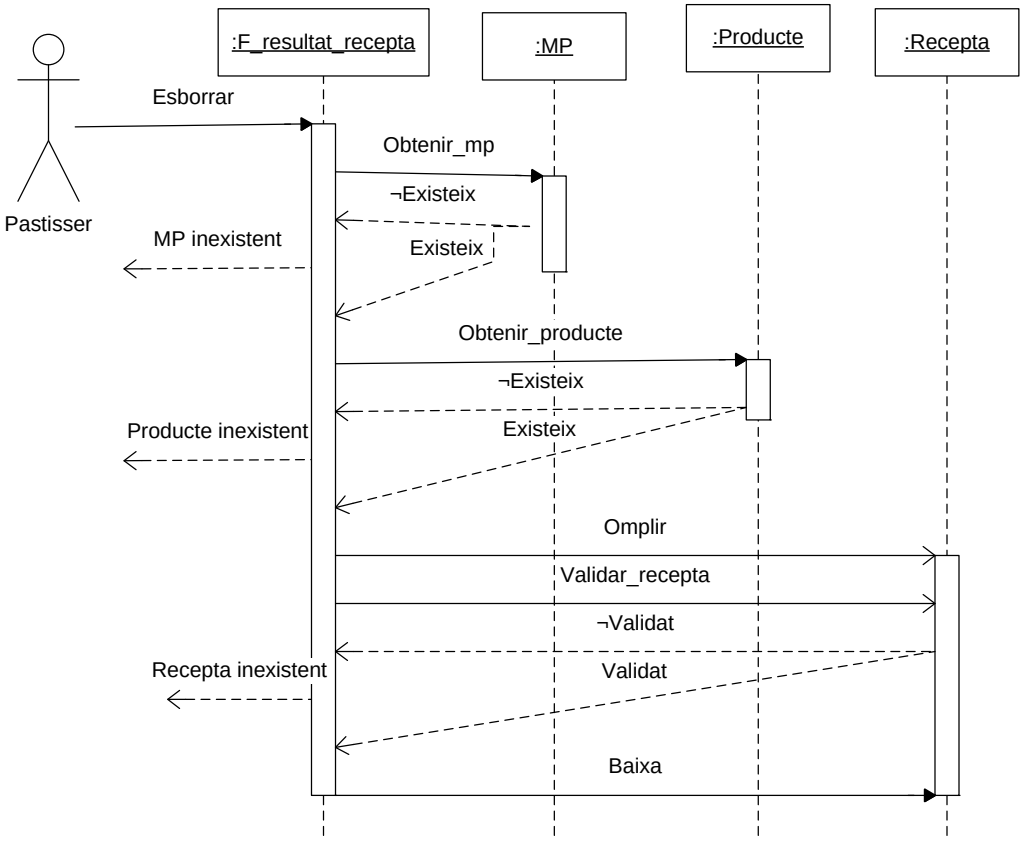


Diagrama seqüència baixa recepta

En aquest esdeveniment hi intervé una classe frontera i tres objectes. En primer lloc, quan es vol donar de baixa una línia d'una recepta s'intenta obtenir la *MP*. En el cas d'aconseguir-ho, s'intenta obtenir el producte i si també es compleix aquesta condició es procedeix a validar si aquella línia encara existeix. Si existeix, es donarà de baixa. En el cas que no es pugui obtenir el *producte*, la *MP* o *validar_recepta*, es mostrarà per pantalla la informació de la situació.

L'últim esdeveniment que falta per acabar aquest cas d'ús és el de **modificar recepta:**

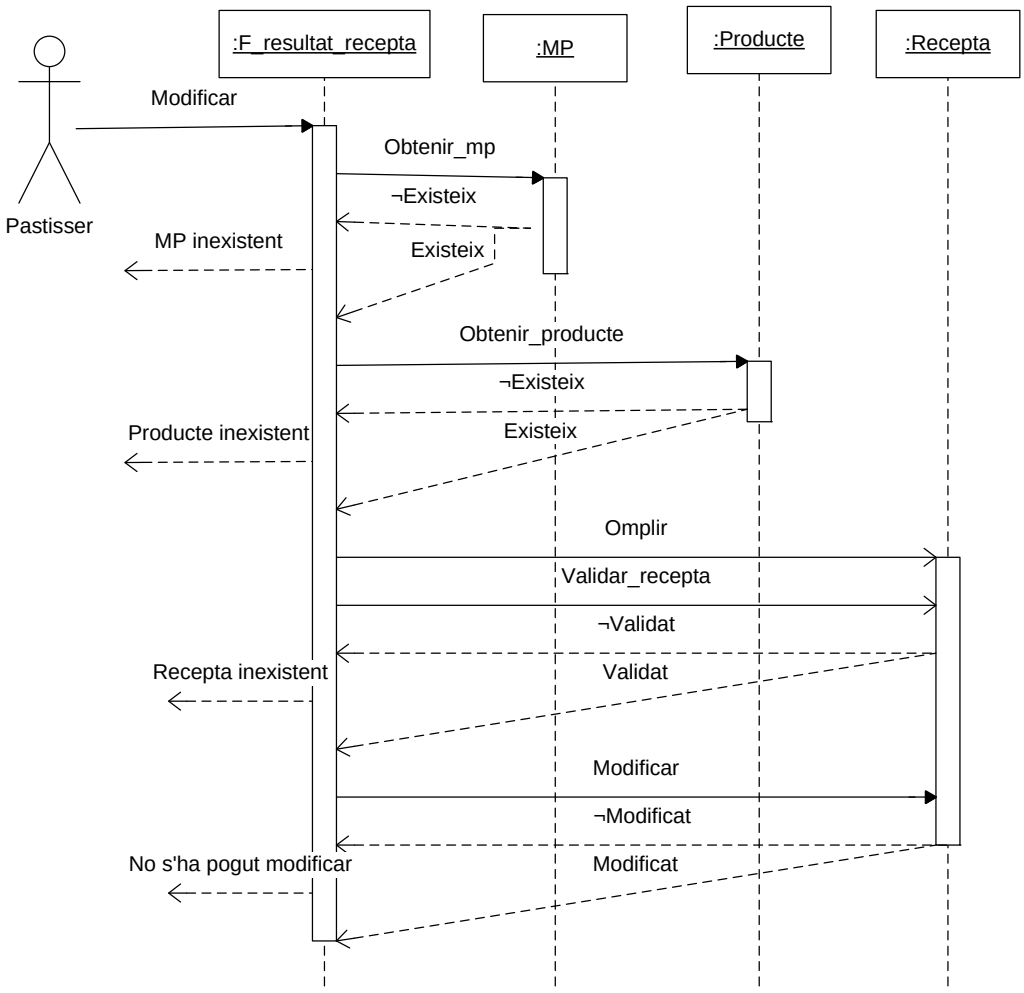


Diagrama seqüència modificar recepta

Aquest esdeveniment és molt similar al de donar de baixa, també necessita una classe frontera i tres objectes. En primer lloc, quan es vol modificar una línia d'una recepta, s'intenta obtenir la *MP.* En el cas d'aconseguir-ho, intenta obtenir el

producte i si també es compleix aquesta condició es procedeix a validar si aquella línia encara existeix. Si existeix, es modificarà amb el nou paràmetre. En el cas que hagi sorgit algun problema en la modificació es mostrarà un missatge per pantalla on s'informarà del problema. En el cas que no es pugui obtenir ni el *producte*, la *MP* o el *validar_recepta*, es mostrarà per pantalla la informació de la situació.

- Diagrama de seqüència de **Manteniment de localitat**

Es començarà per l'**alta:**

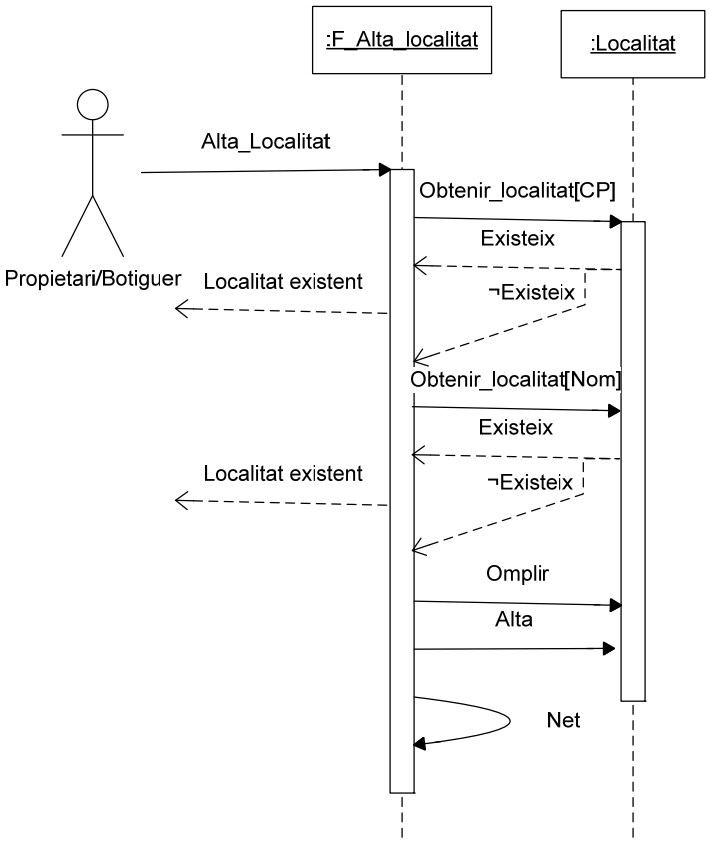


Diagrama seqüència alta localitat

En aquest cas només hi intervé una classe frontera i un objecte. Quan es vol donar d'alta una localitat, en primer lloc, s'intentà obtenir la localitat mitjançant el *CP.* En el cas que no es pugui, s'intentarà obtenir-la pel nom. Si tampoc es pot, es procedirà a omplir l'objecte i, un cop omplert, es donarà d'alta. Si es pot obtenir en qualsevol del dos intents es mostrarà un missatge per pantalla. Per acabar es netejarà la pantalla.

A continuació toca parlar del diagrama de seqüències que pertany a l'esdeveniment **Consulta localitat:**

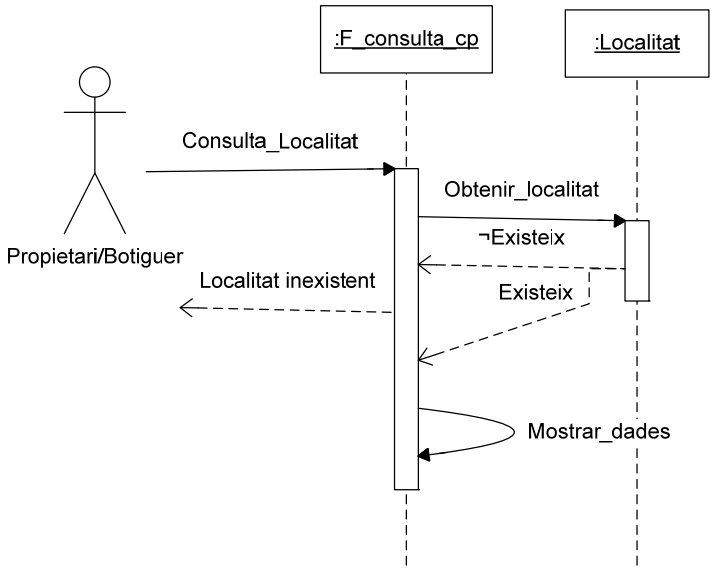


Diagrama seqüència consultar localitat

Quan es vol consultar una localitat, en primer lloc s'intenta obtenir-la. En cas que no es pugui, es mostrarà per pantalla un missatge que ens ho indicarà. En el cas contrari, s'obtindrà la localitat i es mostrarà per pantalla la informació corresponent.

El pròxim esdeveniment a tractar serà el de **baixa de localitat**:

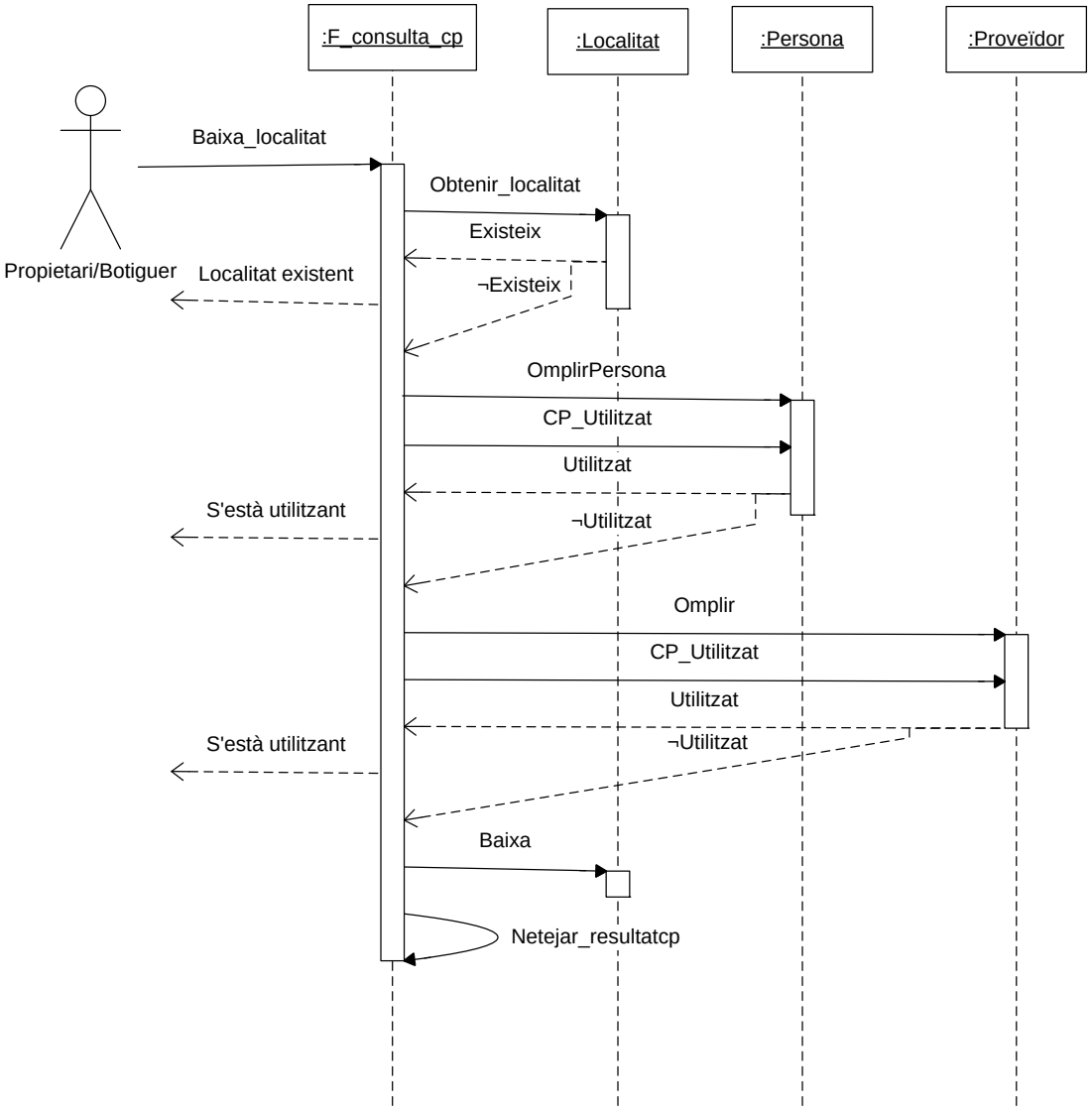


Diagrama seqüència baixa localitat

En aquest esdeveniment hi intervé una classe frontera i tres objectes. En primer lloc, quan es vol donar de baixa una localitat primer s'intenta obtenir l'objecte corresponent. En el cas d'obtenir-lo, es procedeix a omplir una persona qualsevol amb el *CP* corresponent a l'objecte. Un cop omplert, es comprova si s'utilitza o no el *CP* amb la funció *CP_Utilitzat*. En el cas que no sigui utilitzat, es procedirà a realitzar la mateixa operació amb el proveïdor (el funcionament és el mateix que en el cas de la persona) si no s'utilitza, es procedirà a donar de baixa la localitat. En últim lloc, es netejarà la pantalla. En el cas que no es pugi obtenir la localitat o

estigui utilitzada en qualsevol dels dos casos, es mostrarà per pantalla un missatge informant del problema.

Per acabar aquest cas d'ús, es procedirà a mostrar el diagrama de seqüències corresponent a l'esdeveniment **modificar localitat**:

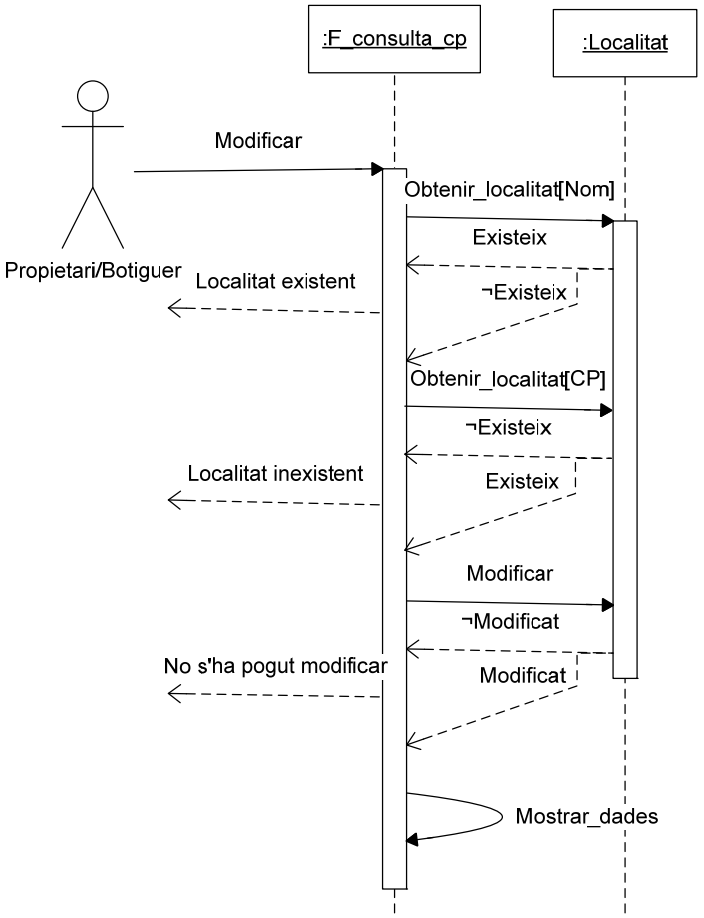


Diagrama de seqüència modificar localitat

En aquest esdeveniment hi intervé una classe frontera i un objecte. Quan es vol modificar una localitat primer s'intenta obtenir mitjançant el nou nom. Si no es pot, significa que el nom d'aquella localitat encara no s'utilitza, per tant, es procedirà a obtenir la localitat mitjançant el *CP*. Si es pot obtenir, ens donarà via lliure per poder modificar-lo. En el cas que el nom ja estigui utilitzat, el *CP* no existeixi o no s'hagi pogut modificar, es mostrarà per pantalla la informació corresponent.

- Diagrama de seqüència de **manteniment de proveïdors**.

Es començarà per l'**alta:**

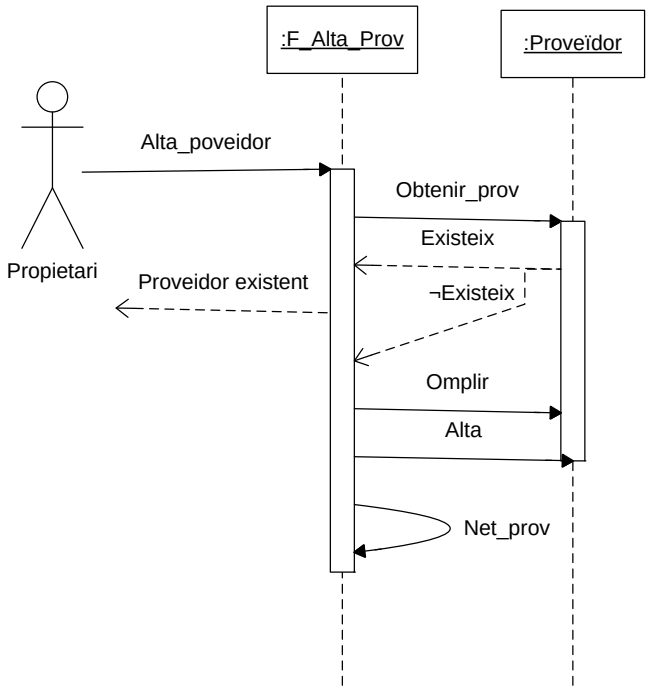


Diagrama de seqüència alta proveïdor

En aquest esdeveniment només hi intervé una classe frontera i l'objecte proveïdor. En primer lloc, s'intenta obtenir el proveïdor, en el cas que no existeixi s'omple l'objecte *proveïdor* i un cop omplert es dóna d'alta. En el cas que es pugui obtenir el proveïdor, es mostrarà un missatge per la pantalla informant del problema.

El següent esdeveniment serà el de **consulta de proveïdors:**

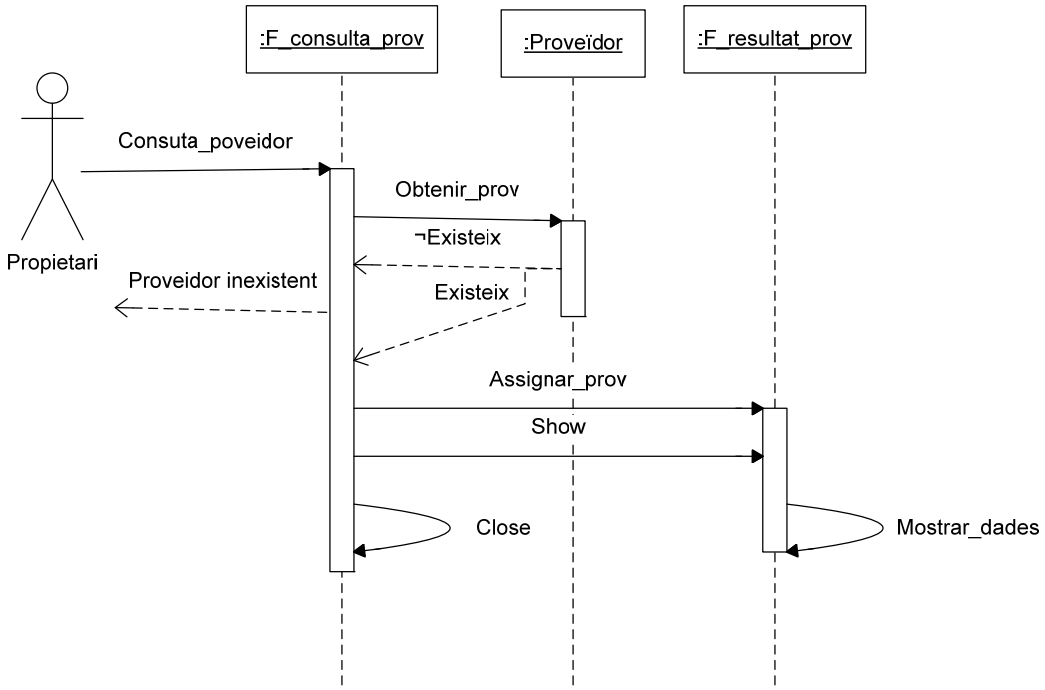


Diagrama seqüència consulta proveïdor

En aquest esdeveniment hi intervenen dues classes frontera i un objecte. En el moment de consultar un proveïdor s'intenta obtenir, en el cas d'aconseguir-ho s'assignarà aquest objecte a la classe frontera *F_resultat_prov*. La classe frontera *F_consulta_prov* mostrarà la nova pantalla amb tota la informació del proveïdor.

El següent esdeveniment serà el de **baixa de proveïdor**:

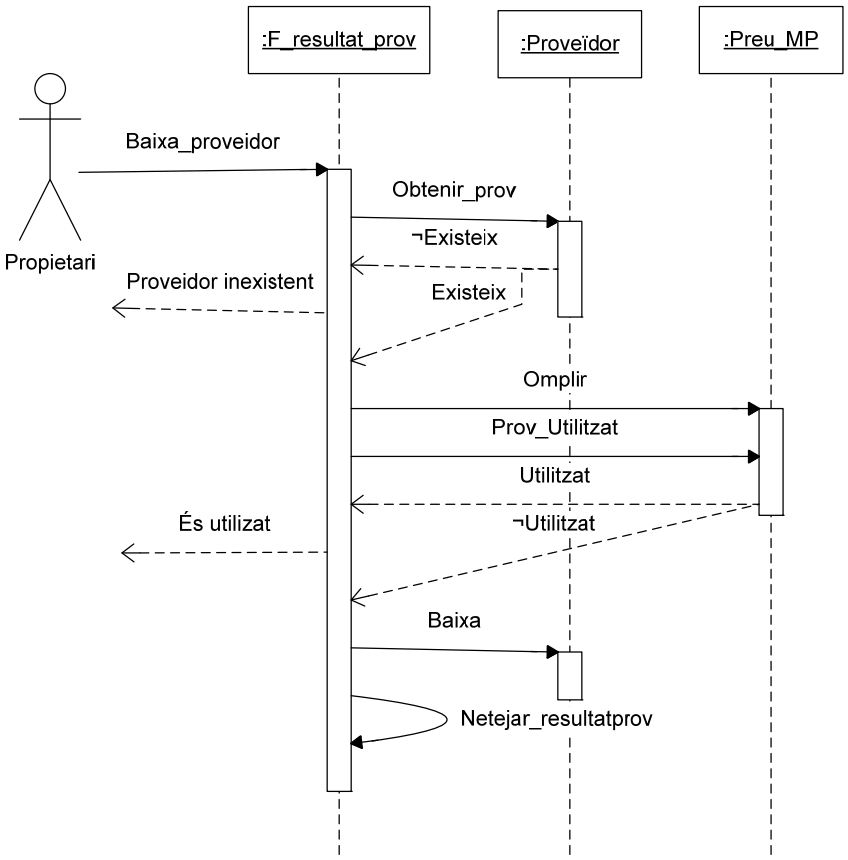


Diagrama de seqüència baixa proveïdor

En aquest esdeveniment hi intervé una classe frontera i dos objectes. En el moment en què l'usuari vol donar de baixa un proveïdor, el sistema realitzarà les següents tasques: en primer lloc intentarà obtenir el proveïdor, si l'aconsegueix, prosseguirà a omplir l'objecte *Preu_MP* (on l'atribut *Id_prod* equivaldrà a l'identificador del proveïdor a esborrar). Un cop l'objecte estigui ple, es buscarà si aquell proveïdor té alguna matèria primera relacionada, en el cas que no en tingui cap, es donarà de baixa. En el cas que no es pugui obtenir o s'utilitzi, es mostrarà per pantalla un missatge informant-ho.

Seguidament mostrarem l'esdeveniment **modificar proveïdor**:

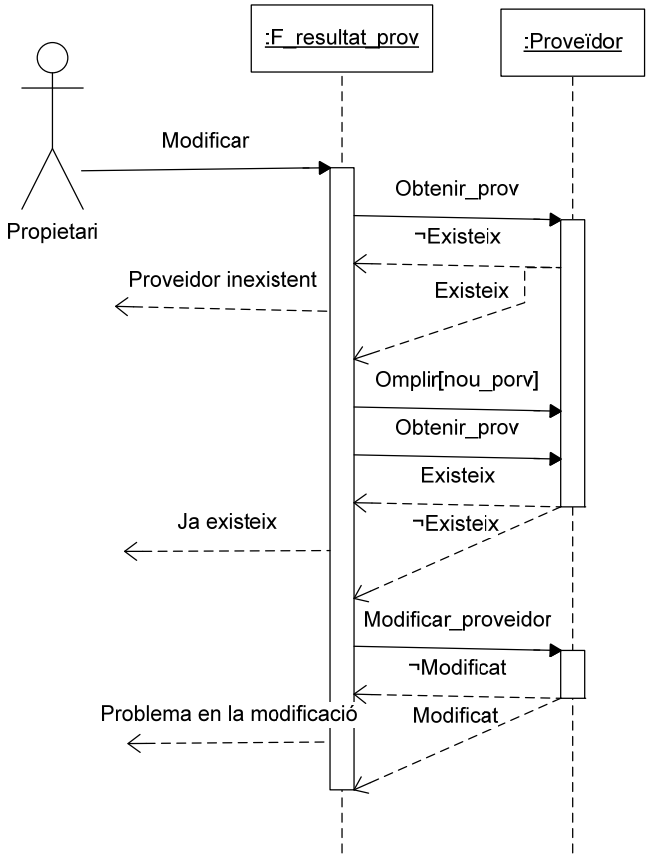


Diagrama seqüència modificar proveïdor

En aquest cas només hi intervé una classe frontera i un objecte. En primer lloc, s'intenta obtenir el proveïdor original. En el cas que es pugui obtenir, es crea un nou objecte amb les noves dades i s'intenta obtenir un proveïdor que tingui el mateix nom. En el cas de no aconseguir-ho, es procedirà a la modificació. Si hi hagués algun problema tan en la obtenció com en la modificació, es mostraria un missatge a la pantalla.

Per acabar aquest cas d'ús, mostraré el diagrama de seqüència corresponent a l'esdeveniment **Assignar preu MP**:

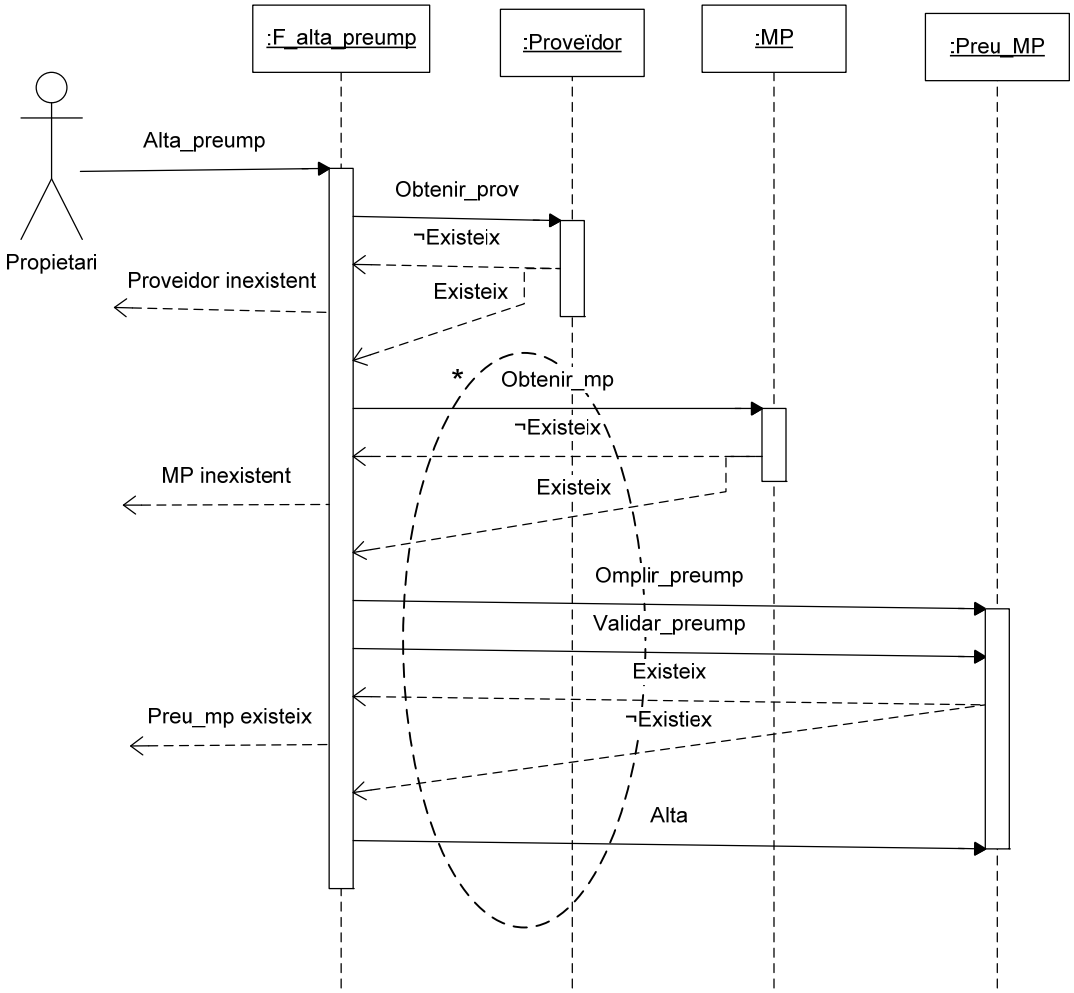


Diagrama seqüència d'assignar preu MP

En aquest esdeveniment hi intervé una classe frontera i tres objectes. En primer lloc, quan es vol assignar a un proveïdor el preu d'una matèria primera, s'intenta obtenir el propi proveïdor. Si ho aconsegueix, s'entra en un bucle amb la funció d'intentar obtenir la matèria primera. Un cop obtinguda, es procedeix a omplir el nou objecte, *Preu_MP,* amb les dades corresponents. Un cop ple es valida si existeix la relació. En el cas negatiu, es procedeix a donar d'alta el *Preu_MP*.

- Diagrama de seqüència de **manteniment de comanda.**

Es començarà per l'**alta:**

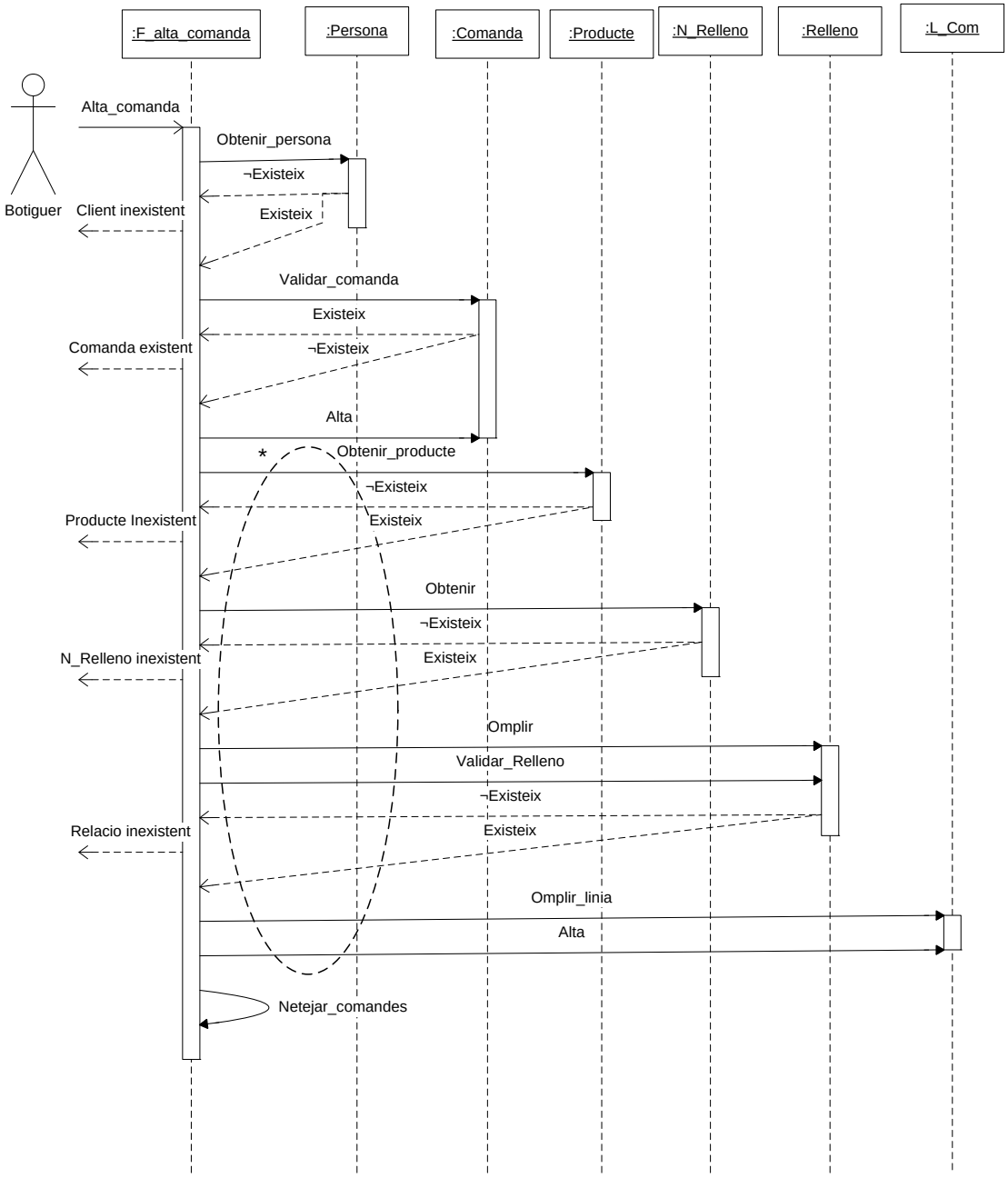


Diagrama seqüència d'alta de comanda

En aquest esdeveniment hi intervé una classe frontera i sis objectes. Quan es realitza l'alta d'una comanda, en primer lloc, s'intenta obtenir el client. Si s'aconsegueix, es valida que no existeixi la comanda. Si tot és correcte, es dóna d'alta la comanda. Un cop creada, entra en un bucle i es procedeix a intentar

obtenir els productes. Si s'obtenen, es procedeix a obtenir el farciment (*N_Relleno*). En cas afirmatiu, s'omple la relació entre el farciment i el producte (*Relleno*) i es valida. Un cop validada, es procedeix a omplir la línia de comanda i es dóna d'alta. Un cop acabat tot el procés, es neteja la pantalla.
En el cas que sorgeixi algun problema en algun dels passos del procediment apareixeran per pantalla missatges informant sobre l'error.

El pròxim esdeveniment a tractar serà el corresponent a la **consulta de la comanda**:

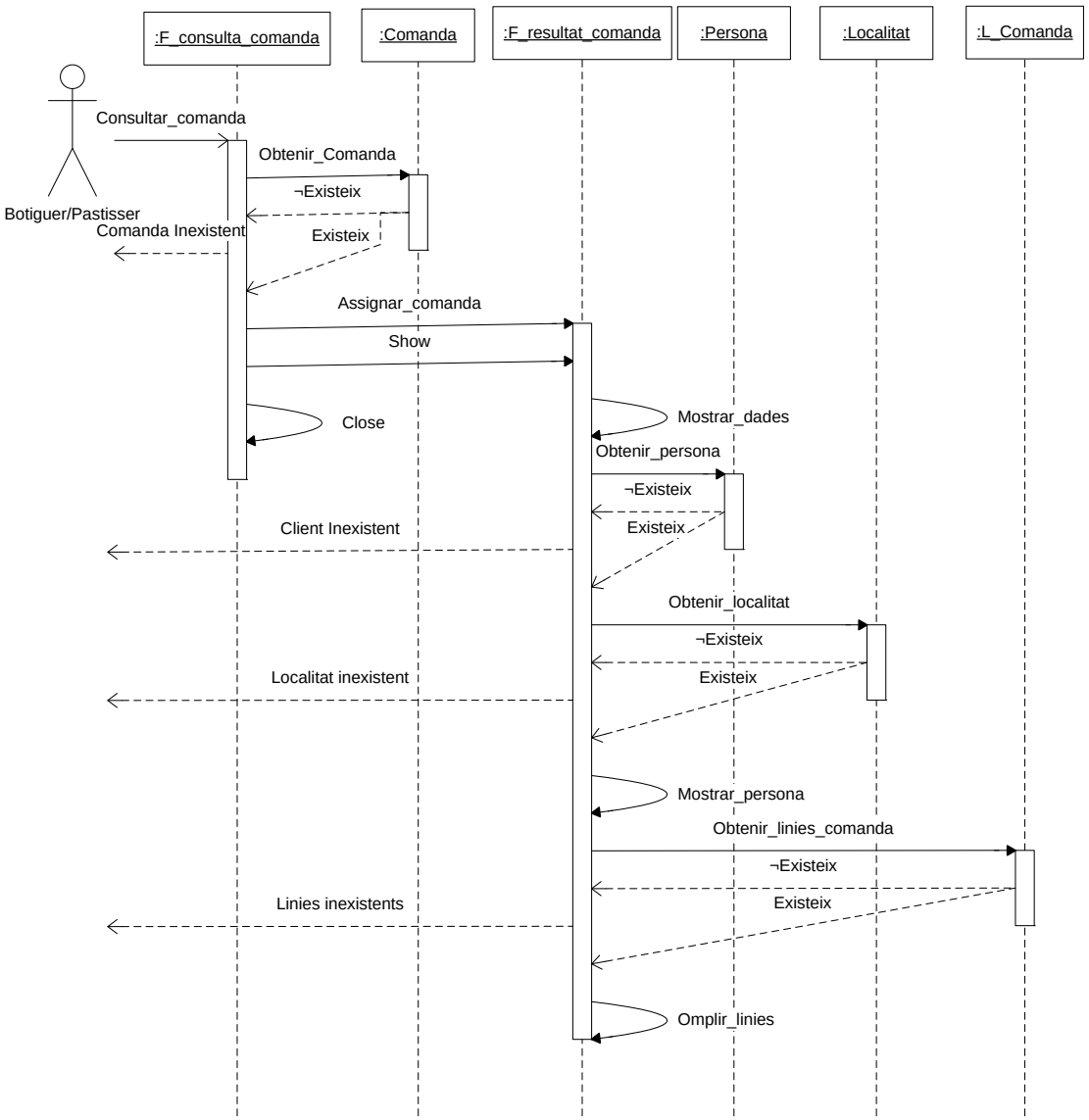


Diagrama de seqüència de consultar comanda

En aquest esdeveniment hi intervenen dues classes frontera i quatre objectes. En primer lloc quan l'usuari consulta una comanda, s'intenta obtenir-la i, si ho aconsegueix, s'assigna a la classe frontera *F_resultat_comanda i es mostra.* Un cop mostrada la nova pantalla, es procedeix a donar les dades de la comanda. Seguidament, s'intentarà obtenir la persona que l'ha realitzada, un cop obtinguda s'intentarà obtenir la localitat d'on és aquest client. Un cop fet, es mostrarà per pantalla la informació de la persona. Després, s'intentaran obtenir les línies corresponents a aquella comanda. Un cop obtingudes, es realitzarà una nova acció encarregada de mostrar-les. En el cas que no es puguin obtenir els diferents objectes esmentats anteriorment, es mostraran per pantalla missatges informant de la situació. A continuació mostraré el diagrama corresponent.

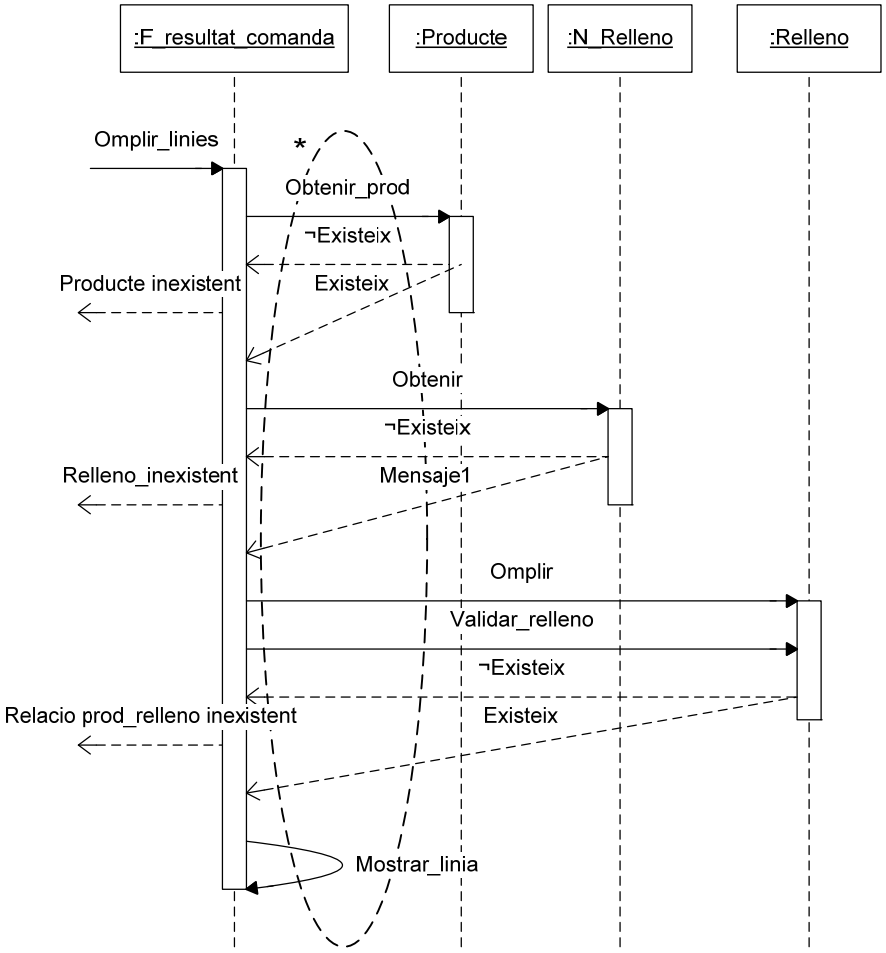


Diagrama de seqüència de l'acció omplir_linies

El procés serà el següent: en primer lloc s'intenta obtenir el producte. Un cop obtingut, s'intenta obtenir el tipus de farciment (*N_Relleno*), un cop obtingut es comprova si la relació entre el producte i el farciment que correspon a l'objecte *Relleno*, és vàlid. En el cas que ho sigui, es mostrarà per pantalla la informació corresponent. Si durant aquest procés sorgeix algun problema s'informarà per pantalla.

El pròxim esdeveniment a tractar, serà la **baixa de comanda:**

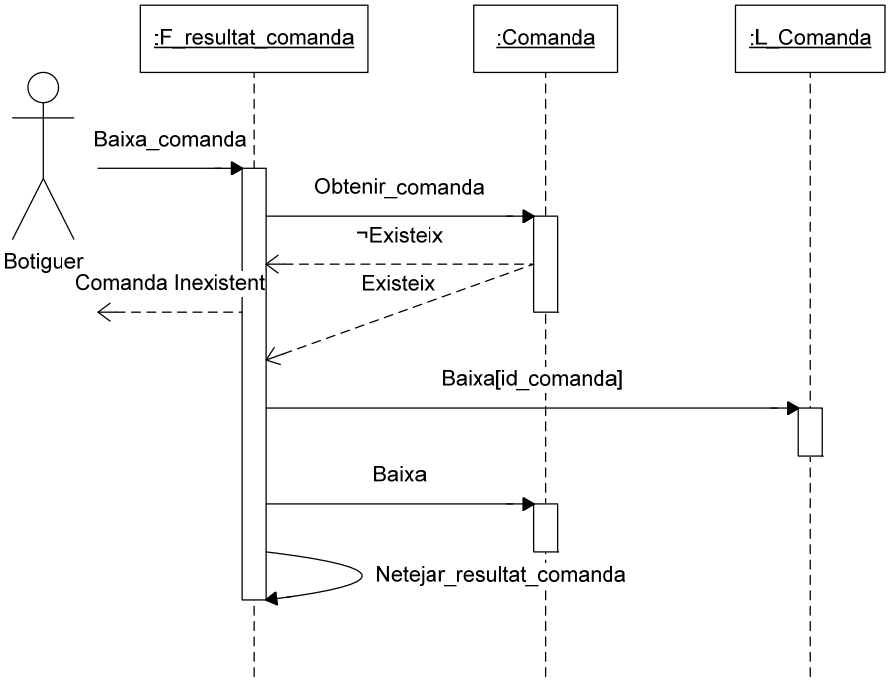


Diagrama de seqüència baixa de comanda

En aquest esdeveniment només hi intervenen dos objectes i una classe frontera. En el moment de donar de baixa una comanda, s'intenta obtenir la capçalera de la comanda. Quan s'aconsegueix, es procedeix a eliminar les línies de comades amb la acció *baixa* que se l'hi ha de passar per paràmetre l'identificador de la comanda. Un cop esborrades totes les línies, es procedeix a esborrar la capçalera de la comanda amb l'acció *baixa*. En el cas que no es pogués obtenir la comanda, es mostraria per pantalla la informació corresponent.

També ens pot interessar donar de baixa alguna línia determinada en comptes de tota la comanda, a continuació es mostrarà el diagrama corresponent.

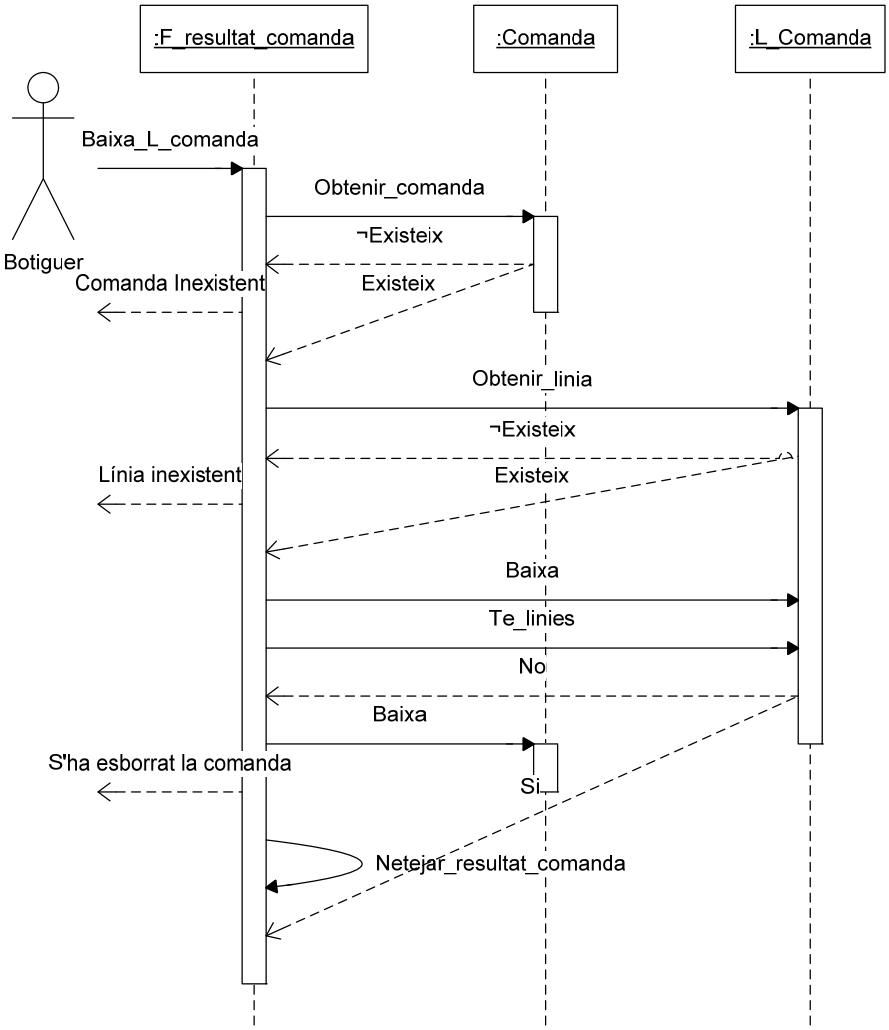


Diagrama seqüència baixa línia comanda

Tot aquest procediment s'utilitza per poder donar de baixa una línia determinada. En primer lloc, s'intenta obtenir la comanda, en el cas d'aconseguir-ho, es procedirà a obtenir la línia. Quan s'hagi obtingut, es procedirà a donar-la de baixa. En aquest cas no serà necessari passar-li l'identificador de la comanda. Un cop esborrada, es comprovarà si aquella comanda encara té línies associades. Si ja no en queda cap, s'esborrarà també la capçalera de la comanda i es netejarà la pantalla. En el cas contrari no s'esborrarà res. Si durant tot el procés sorgeix algun problema, es mostrarà per pantalla un missatge.

L'últim esdeveniment que falta per acabar aquest cas d'ús és el de **modificar la comanda**.

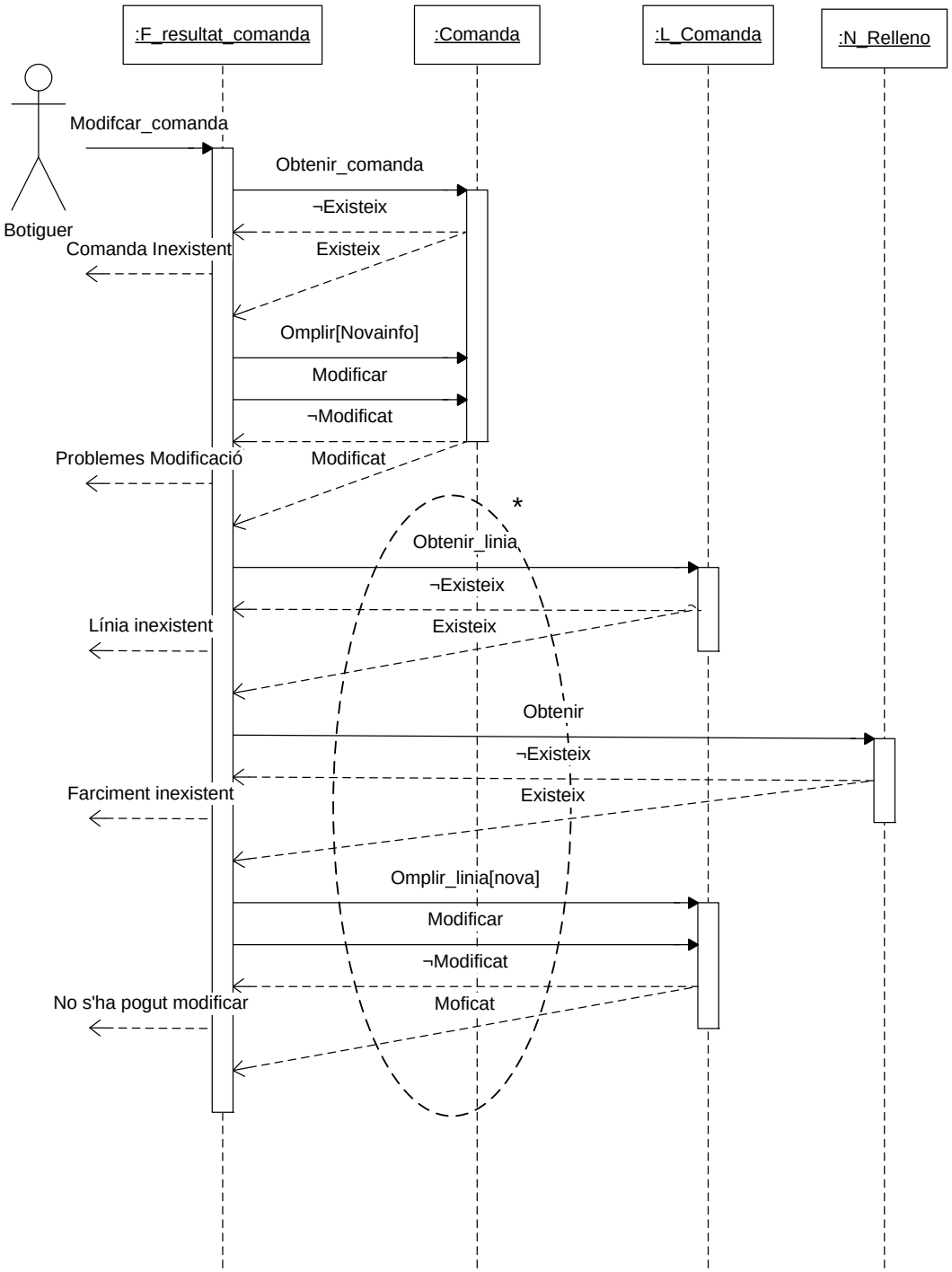


Diagrama seqüència modificar comanda

Quan l'usuari vol modificar una comanda, la primera cosa que fa el sistema és intentar obtenir la comanda. Si ho aconsegueix, es procedeix a omplir un altre objecte amb les noves dades de la comanda i es modifica la capçalera. Si tot ha anat bé, el sistema començarà un bucle en què s'intentaran obtenir les diferents línies. Un cop obtingudes, s'intentarà obtenir el farciment (*N_Relleno*). Un cop obtingut, s'omplirà una nova línia de comandes i es procedirà a modificar la línia original. D'aquesta manera, es modificaran també la resta de línies. En el cas que sorgeixi algun problema en el transcurs de tot el procés, es mostrarà per pantalla la informació pertinent.

- Diagrama seqüència de **manteniment de farciment**.

Es començarà per l'**alta**:

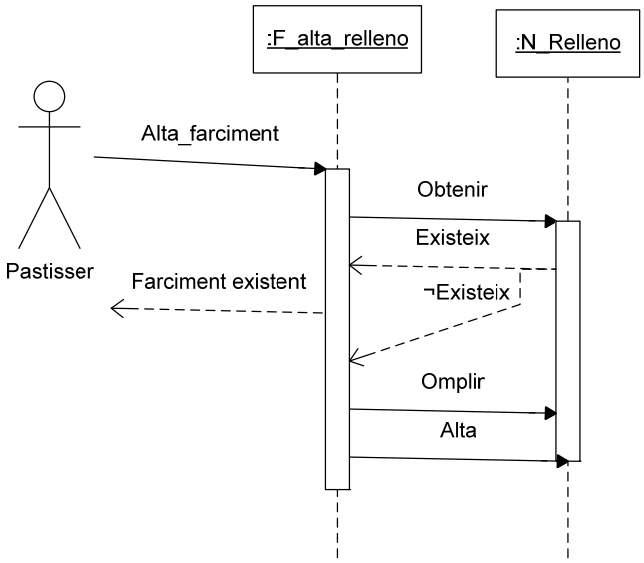


Diagrama de seqüència d'alta de farciment

En primer lloc, quan es vol donar d'alta un farciment, el sistema intenta obtenir-lo mitjançant el nom. En el cas que no es pugui obtenir, s'omplirà el nou objecte amb les dades corresponents i es prosseguirà a donar-lo d'alta. Si es pogués obtenir el farciment, es mostraria per pantalla un missatge informant-ho.

El pròxim esdeveniment que es tractarà, serà el de **consultar el farciment**:

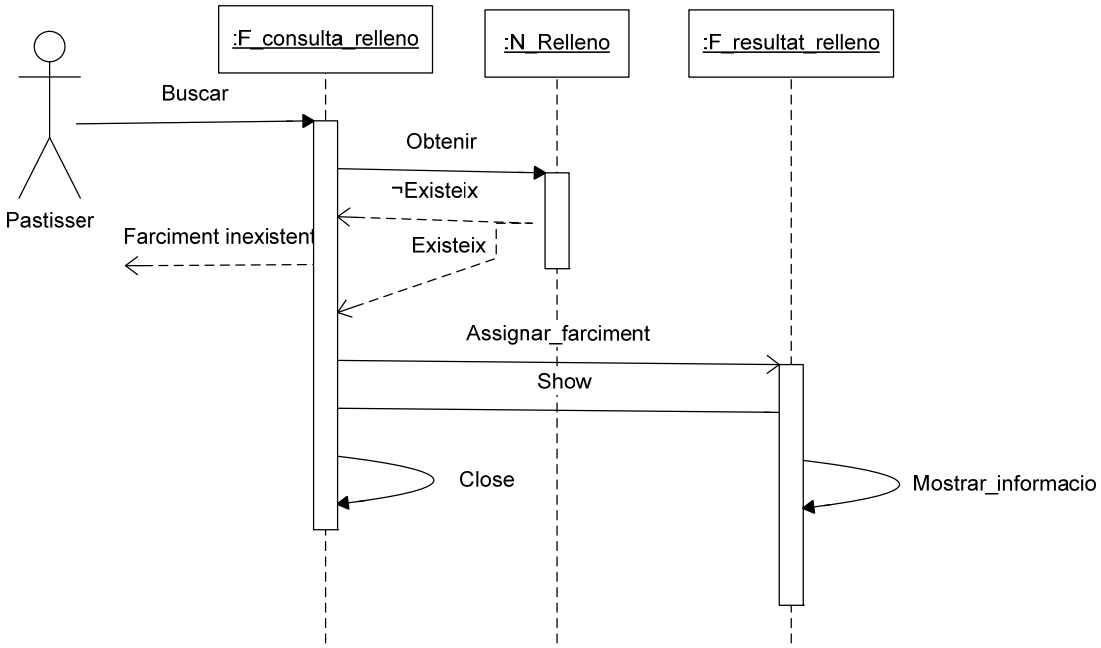


Diagrama de seqüència de consultar farciment

En primer lloc, quan es vol consultar, s'intenta obtenir el Farciment (*N_Relleno*). Un cop s'ha obtingut, s'assigna aquest objecte a la classe frontera *F_resultat_relleno* i es mostra per la pantalla. En últim lloc, es mostren les dades corresponents al farciment consultat. Si hi ha algun problema en el procés s'informarà amb un missatge.

El pròxim esdeveniment a tractar serà la **baixa del farciment**:

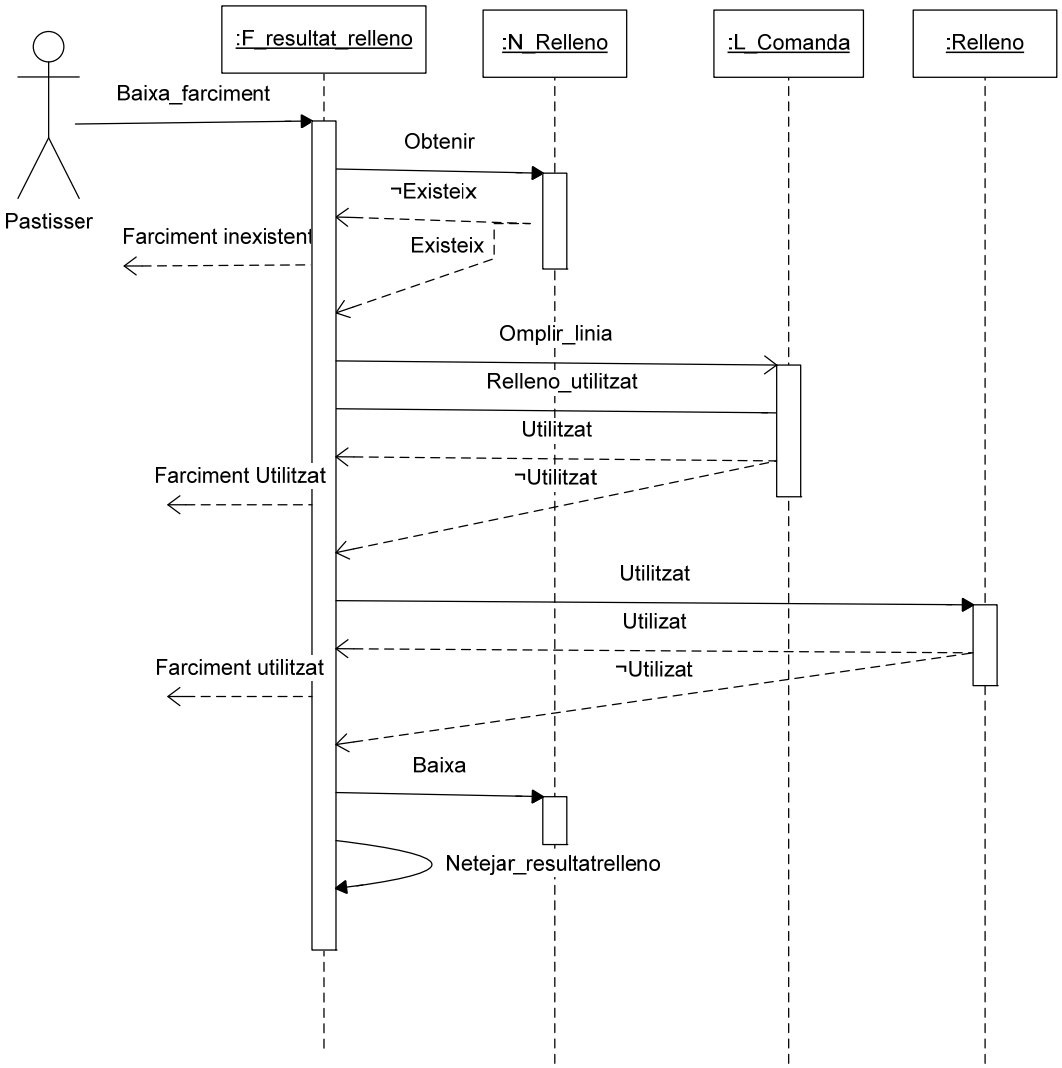


Diagrama de seqüència de baixa de farciment

En aquest esdeveniment hi intervé una classe frontera i tres objectes. En primer lloc, quan es vol donar de baixa un farciment, s'intenta obtenir-lo. Un cop obtingut, es procedeix a omplir una línia de comanda que ens farà la funció de màscara ja que només ens interessarà l'atribut *farciment*. Un cop omplerta aquesta línia, es comprovarà si s'està utilitzant en alguna comanda aquell farciment. En el cas negatiu, es realitzarà la mateixa tasca amb l'objecte *Relleno.* També s'omplirà en forma de màscara i es comprovarà si algun producte l'utilitza, en el cas que no sigui utilitzat, es procedirà a donar de baixa el farciment.

El següent esdeveniment que es tractarà, serà el de **modificar el farciment**:

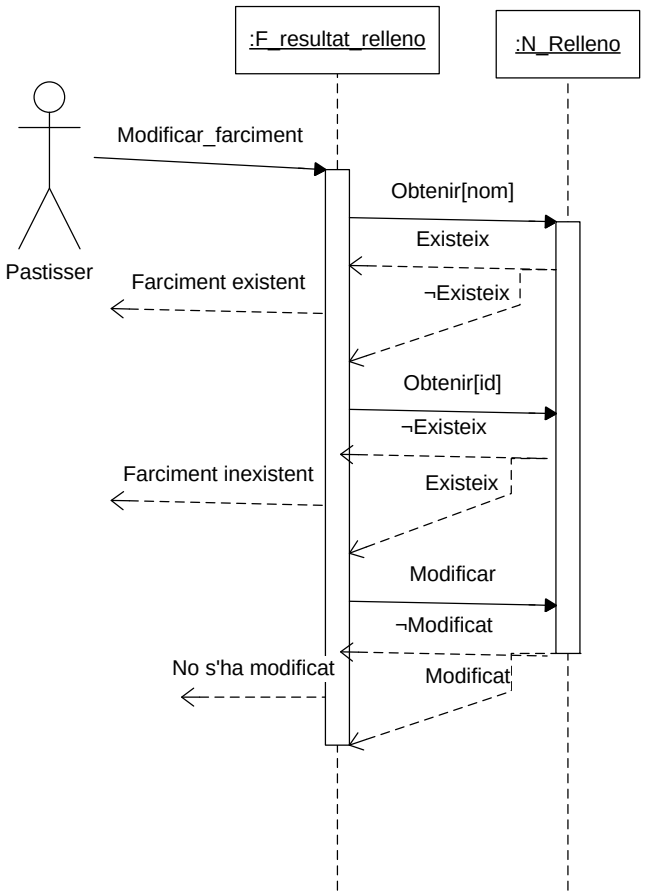


Diagrama de seqüència modificar farciment

En aquest esdeveniment hi intervé una classe frontera i un objecte. Per modificar un farciment, en primer lloc, s'intenta obtenir el nom del nou farciment. En el cas que no existeixi aquell nom, es procedirà a obtenir el farciment a modificar. Si s'ha pogut obtenir, es procedirà a modificar l'objecte utilitzant l'objecte creat. En el cas que sorgeixi algun problema en aquest procés, per pantalla es mostrarà la informació.

Per acabar aquest cas d'ús, es mostrarà el diagrama de seqüència corresponent a l'esdeveniment **assignar farciment:**

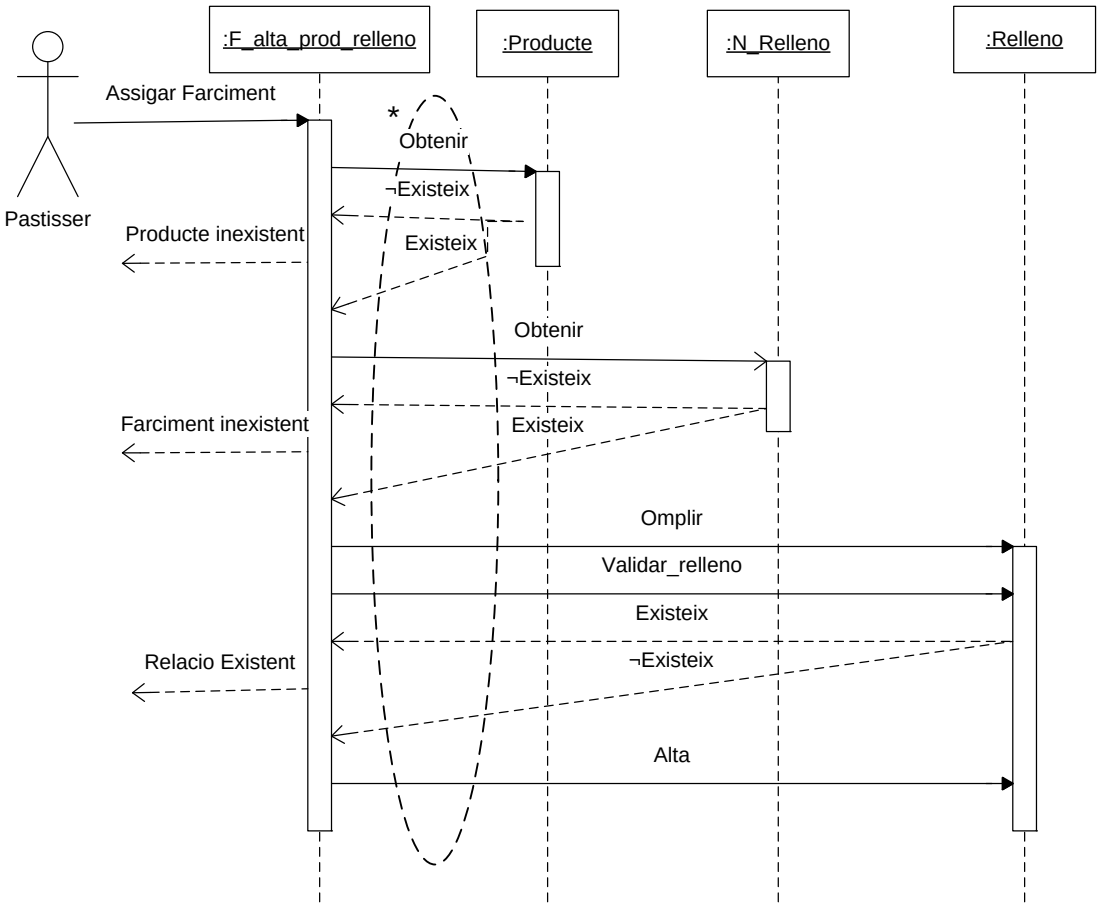


Diagrama de seqüència d'assignar farciment

En aquest esdeveniment hi intervé una classe frontera i tres objectes. Quan es vol assignar a un producte un farciment els passos a seguir són els següents: en primer lloc, s'intenta obtenir el producte i un cop obtingut, es procedeix a obtenir el farciment. Quan s'ha obtingut, es procedeix a omplir l'objecte *Relleno* i es procedeix a validar si aquella relació entre producte i farciment existeix o no. En el cas que no existeixi es procedirà a donar d'alta la relació.

- Diagrama de seqüència del **TPV**:

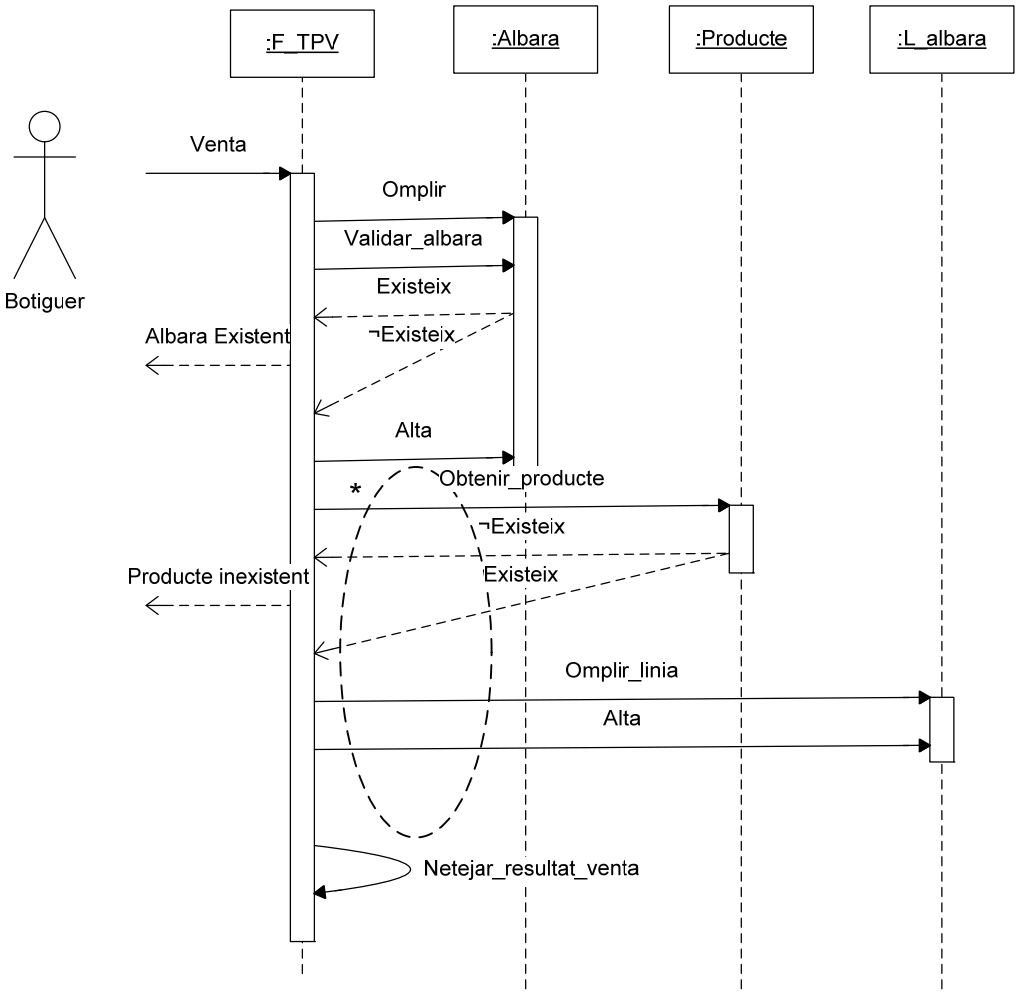


Diagrama de seqüència del TPV

En aquest esdeveniment hi intervé una classe frontera i tres objectes. Quan es realitza una venda, en primer lloc s'omple la capçalera i es valida si l'albarà existeix o no. Si no existeix, entrarà en un bucle que consisteix a intentar obtenir el producte. Un cop obtingut, s'omplirà una línia de l'albarà amb les dades corresponents i es donarà d'alta. En el cas que sorgeixi algun problema en el procés es mostrarà per pantalla un missatge informant-ho.

- Diagrama de seqüència de **Calcular MP per comandes:**

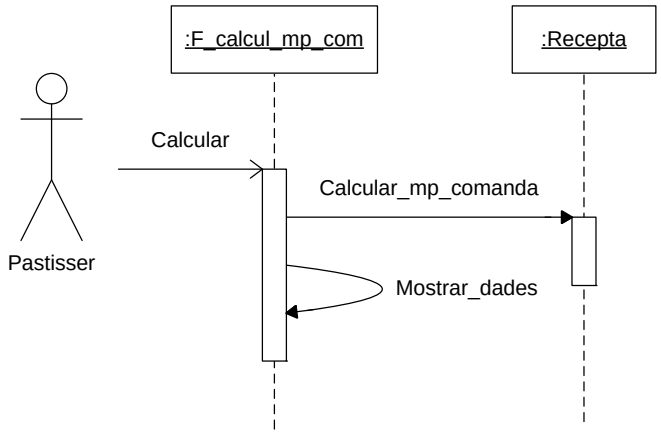
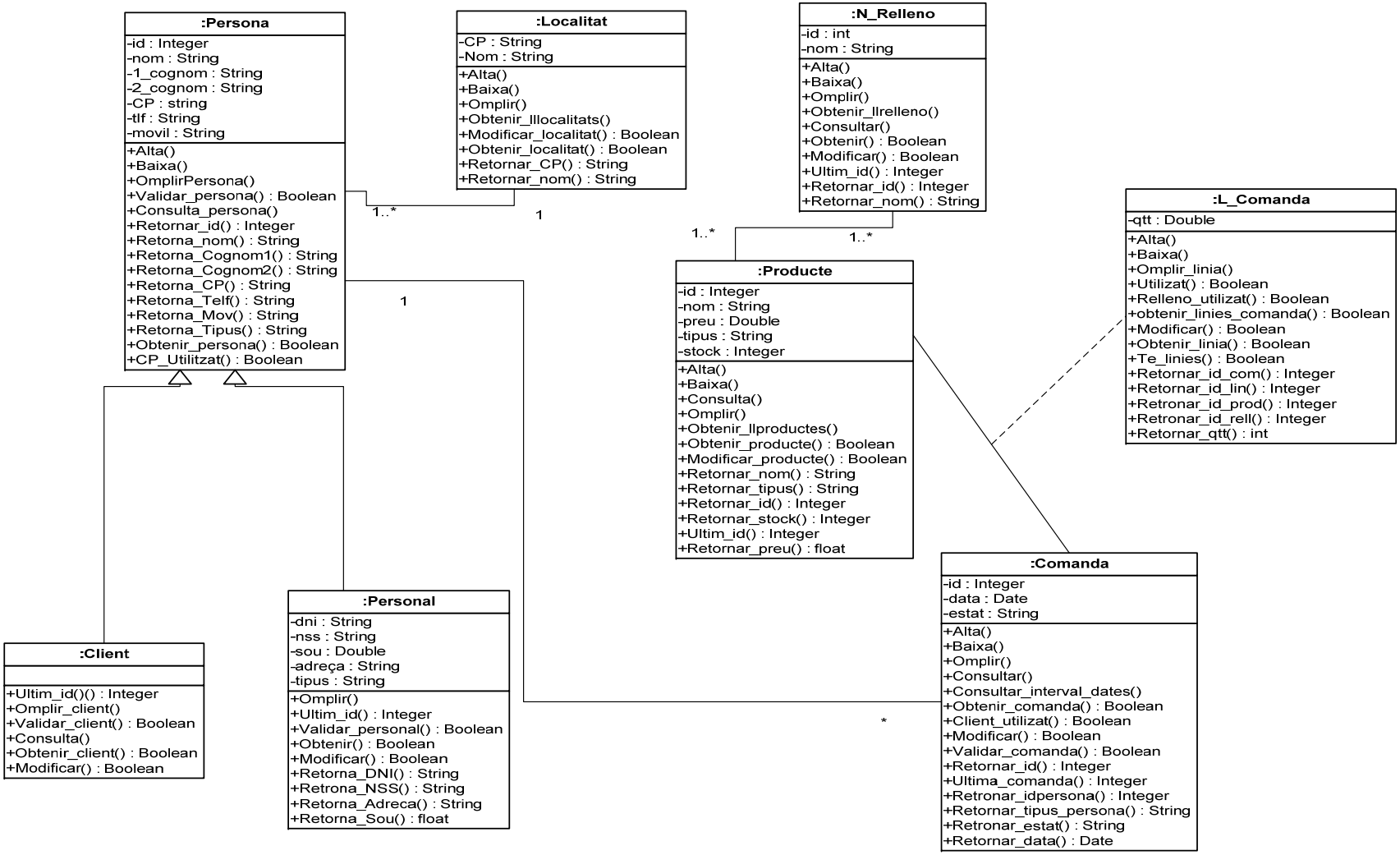


Diagrama de seqüència de calcular MP per comanda

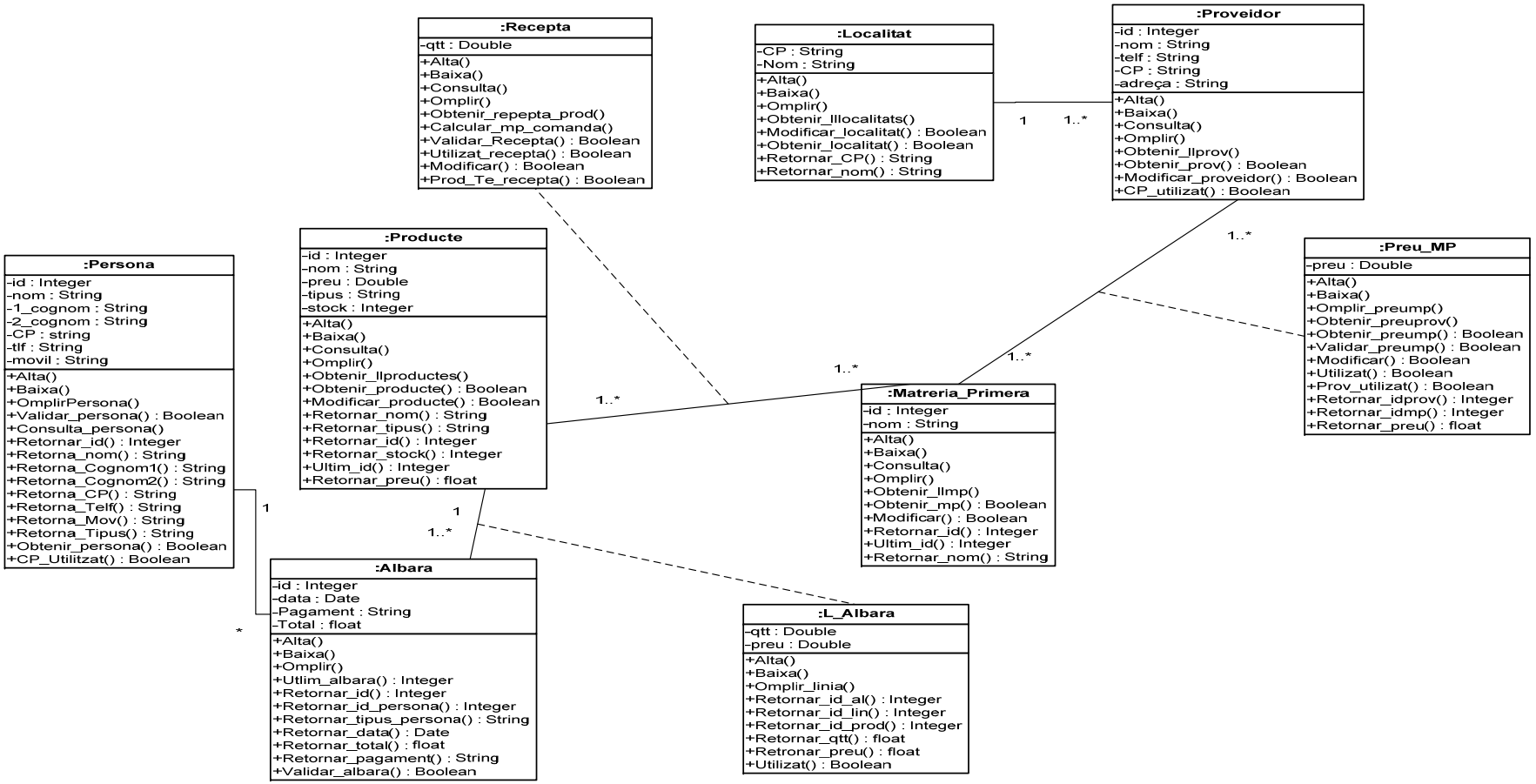
En aquest esdeveniment hi intervé només una classe frontera i un objecte. Quan es vol obtenir el llistat, la classe frontera envia l'esdeveniment *Calcular_mp_comandes* que retrona el llistat de les *Matèries_primeres* necessàries per satisfer l'interval de comandes.

3.3. Ampliació del diagrama de classes

Un cop vistos tots els diagrames, ampliarem el diagrama de classes inicial afegint-hi les operacions de cada classe, que són els diferents missatges que envien.
Com que l'ampliació no cap en una sola plana, dividiré el diagrama de classes en dos:



Ampliació del diagrama de classes



Ampliació del diagrama de classes

Una de les propietats principals de la orientació a objectes és l'herència, per tant, a continuació es mostrarà un diagrama de classes corresponent a les finestres i l'herència que hi ha entre elles.

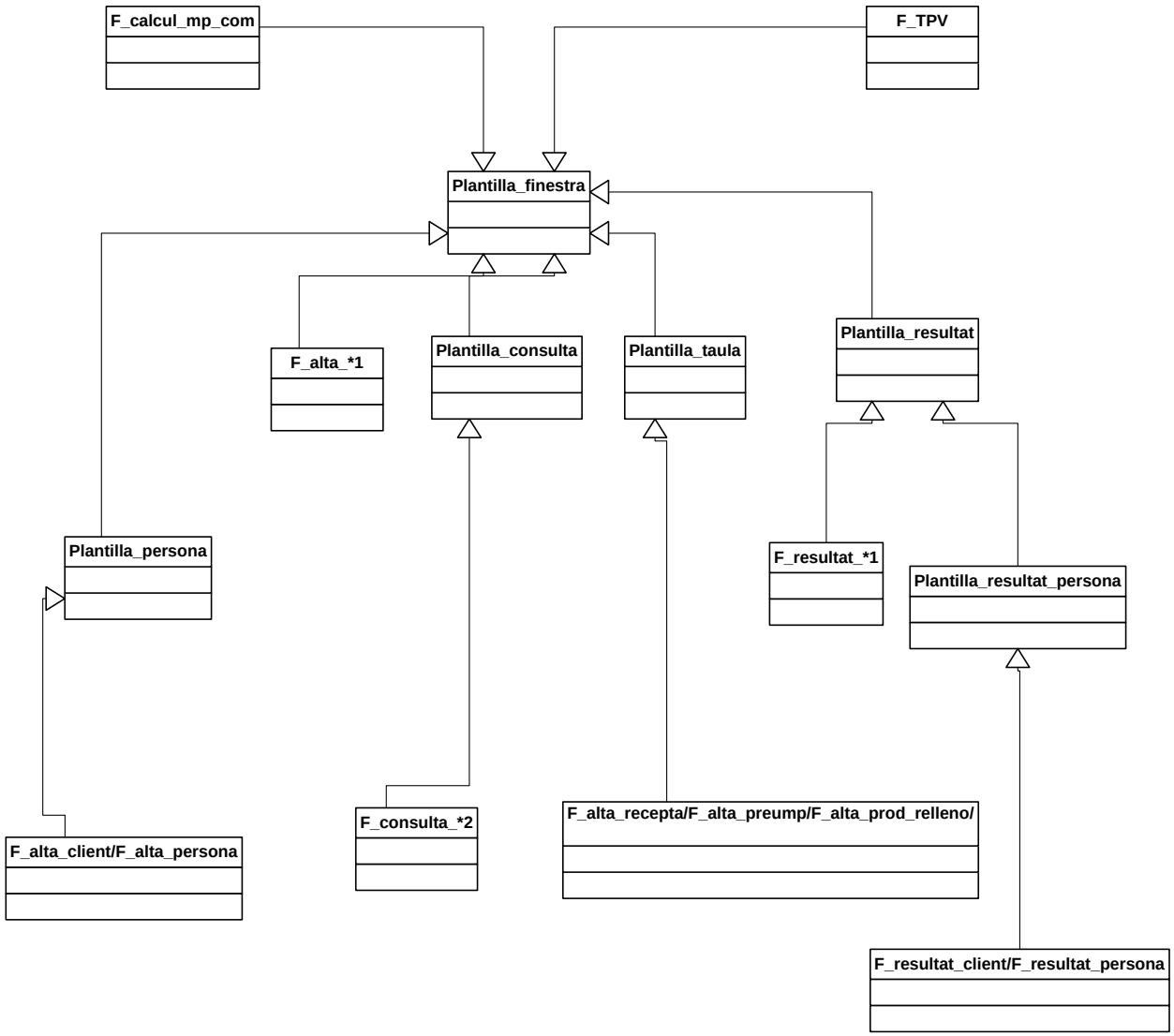
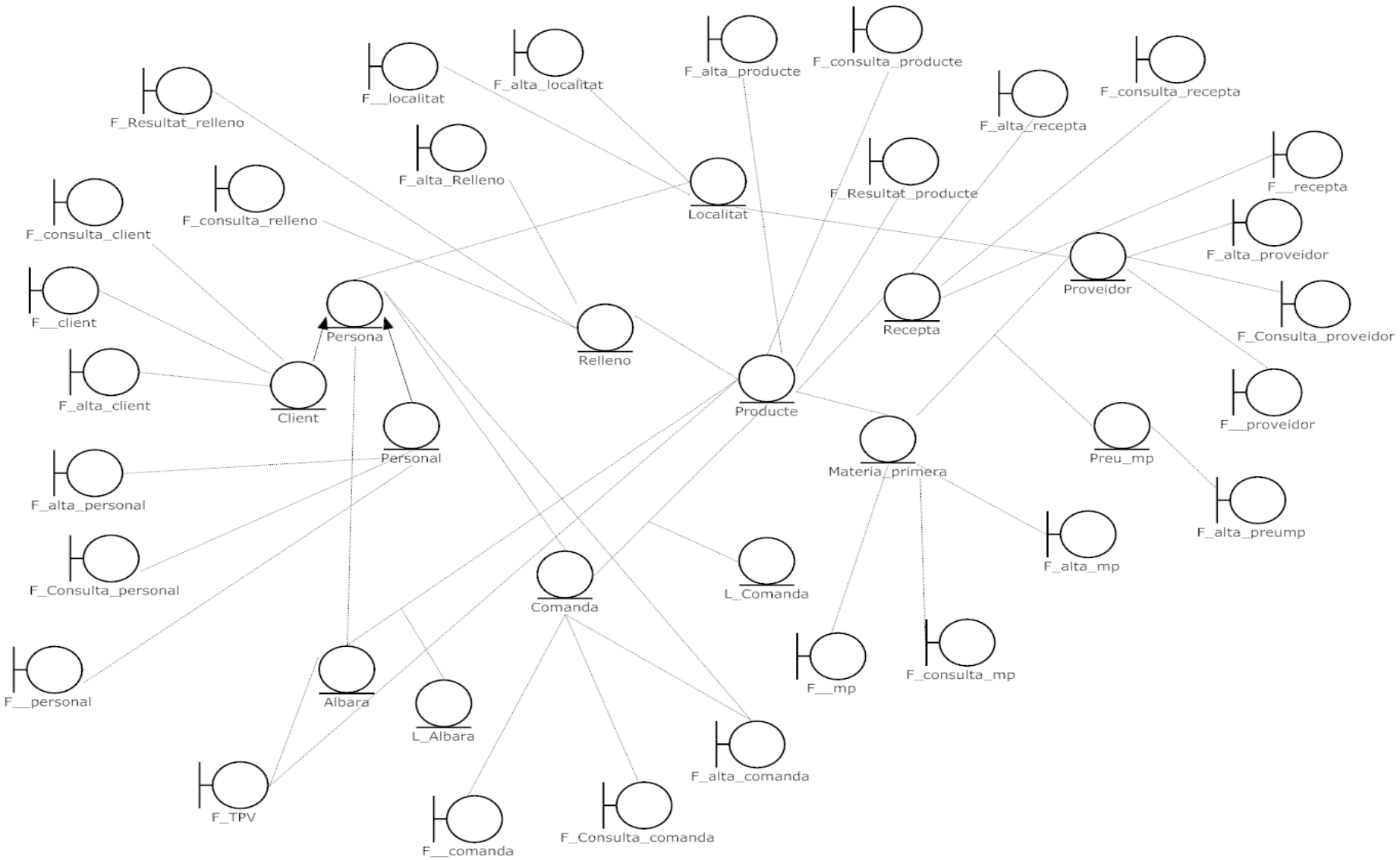


Diagrama de classes de les interfícies

El símbol *1 correspon als següents objectes: **Producte, MP, Proveïdor, farciment, localitat i comanda.**

Per acabar aquesta part, ampliarem un altre cop el Diagrama de classes original de manera abstracta, és a dir, sense que hi figurin els atributs ni les operacions però que incloguin les diferents **classes frontera** que ens han sortit.


F_Resultat_relleno
F__localitat
F_alta_localitat
F_alta_producte
F_consulta_producte
F_alta_recepta
F_consulta_recepta
F_alta_Relleno
F_Resultat_producte
F__recepta
F_consulta_client
F_consulta_relleno
Localitat
F_alta_proveidor
F__client
Persona
Relleno
Recepta
Proveidor
F_Consulta_proveidor
F_alta_client
Client
Producte
F__proveidor
Personal
Materia_primera
Preu_mp
F_alta_personal
F_alta_preump
F_Consulta_personal
Comanda
L_Comanda
F_alta_mp
F__personal
Albara
F__mp
F_consulta_mp
L_Albara
F_alta_comanda
F_TPV
F__comanda
F_Consulta_comanda

4. Disseny del sistema

En aquest apartat, s'han de lligar els diagrames analitzats en els anteriors apartats a una tecnologia concreta. L'etapa de disseny consta de diverses fases:

1. *Disseny arquitectònic*: en aquest apartat, s'esmenta la manera com es distribuirà el software desenvolupat sobre l'entorn on l'executarem.
2. *Disseny d'interfícies:* es detalla la part d'interacció que hi ha entre el sistema i l'usuari.
3. *Disseny de la Base de dades*: un cop s'ha vist el diagrama **Entitat-Relació** es traduirà per poder implementar-lo en SQL.
4. Disseny del programa: es mostraran exemples del codi del propi programa.
5. Pàgina web: en aquest apartat s'explicarà el funcionament de la pàgina web i la seva protecció.
6.

4.1. Disseny arquitectònic

Actualment, l'empresa disposa només d'un ordinador però no es descarta que en un futur en puguin tenir un altre. Per tant, es farà el disseny arquitectònic pensant en els dos sistemes. A més, l'empresa desitja que el programa sigui el mateix perquè l'usuari el pugui utilitzar en qualsevol dels dos ordinadors.

Per realitzar aquest apartat, s'elaboraran dos diagrames:

- **Diagrama de components:** mostra com agrupem el *software* a desenvolupar.
- **Diagrama de desplegament:** defineix l'entorn on ha d'executar-se el software i la manera com es distribueixen els components.

Diagrama de components

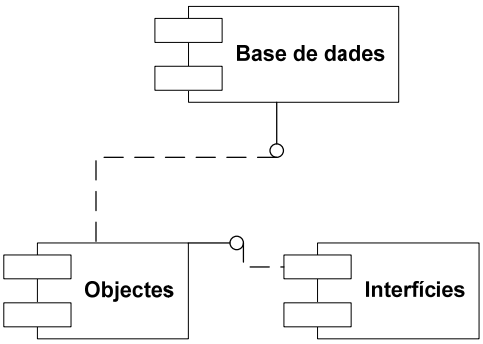


Diagrama de components

En aquest diagrama es mostra la manera com s'han agrupat els diferents components. A continuació, es desglossarà la relació que hi ha entre la interfície i l'objecte. En primer lloc, però, es mostrarà com s'han agrupat les interfícies.

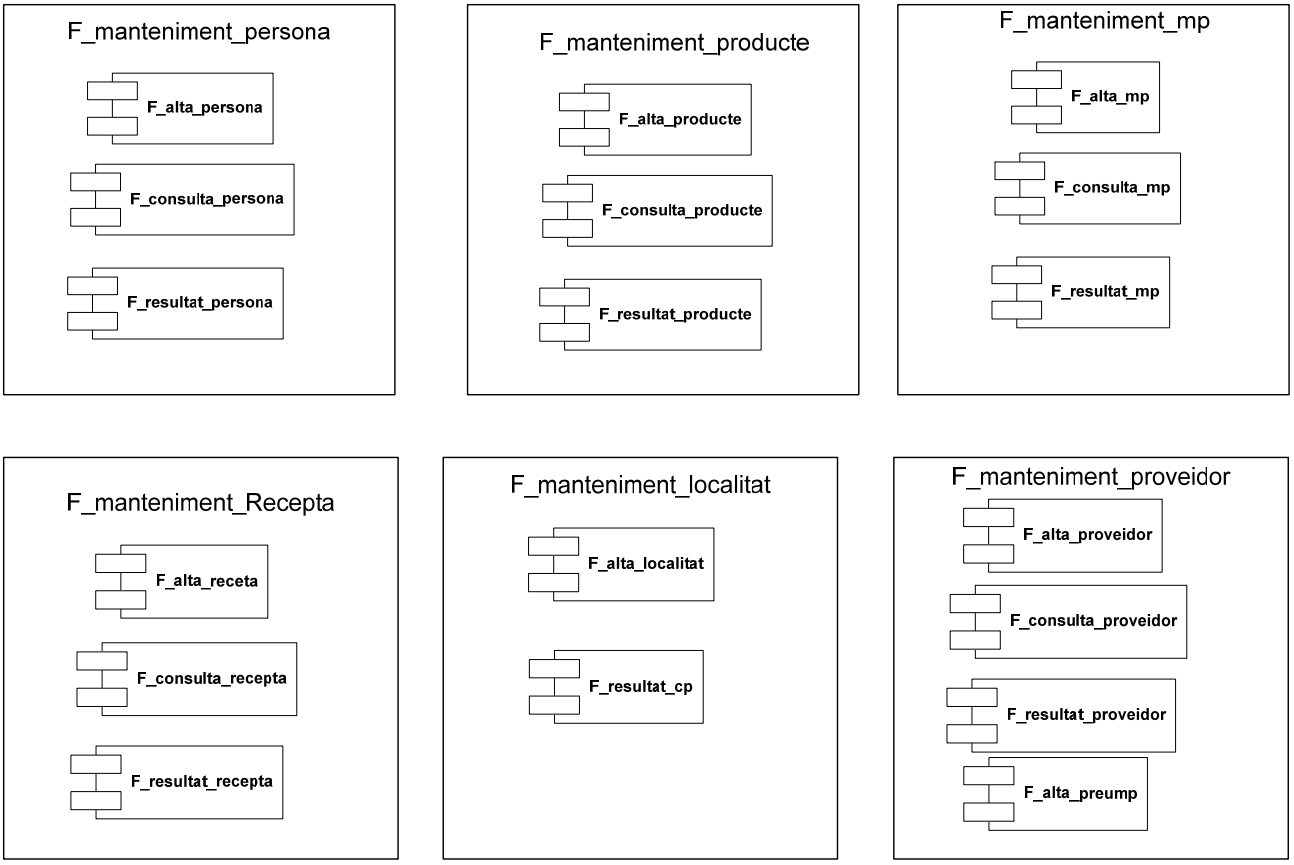


Diagrama components

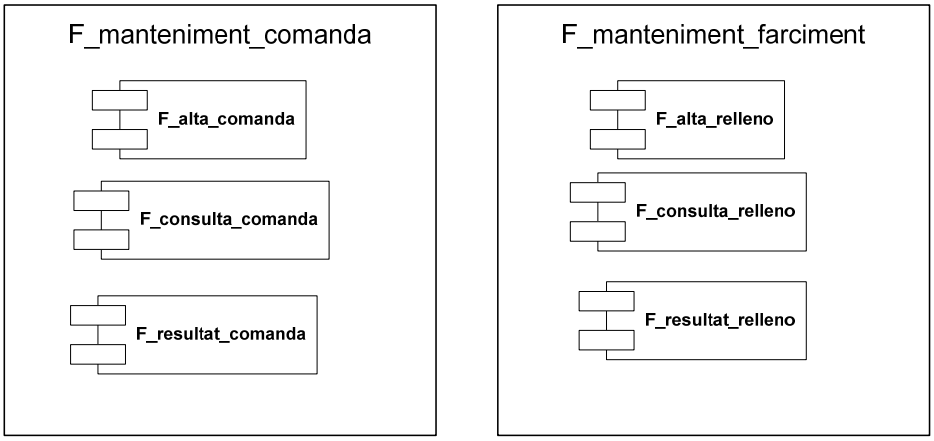


Diagrama components

A continuació, es mostrarà la manera com estan relacionades aquestes interfícies amb els objectes que utilitzen.

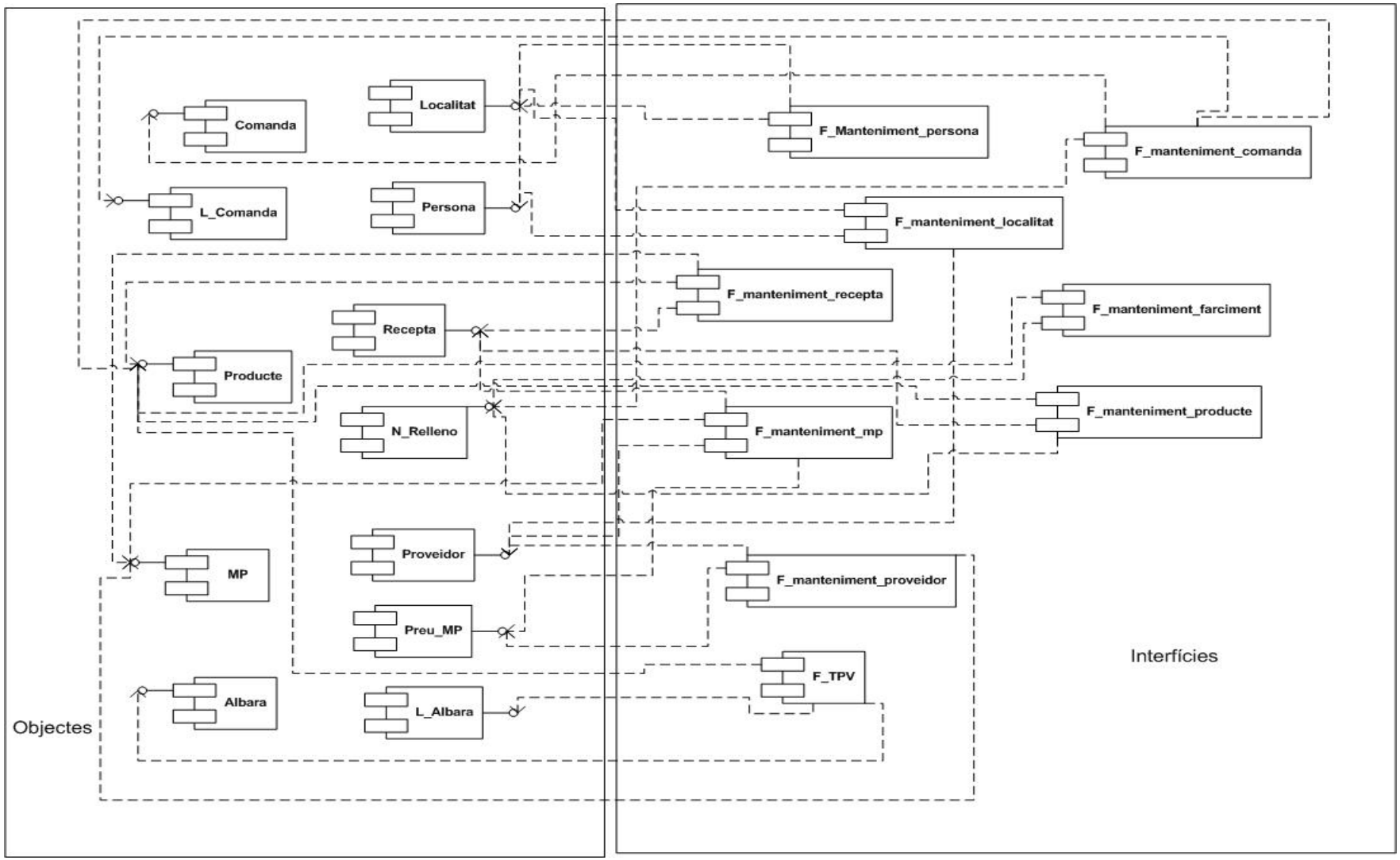


Diagrama de components

Diagrama de desplegament

Com ja s'ha comentat anteriorment, es disposa de dos clients. Es vol que cadascun d'ells tingui la totalitat del programa. Per tant, la disposició del *software* en el *hardware* pot ser semblant a la que es mostra a continuació:

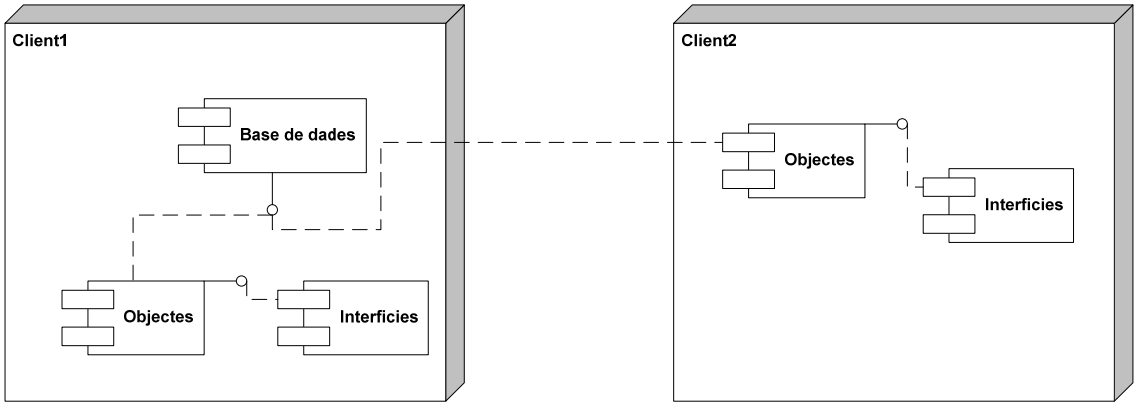


Diagrama de desplegament

En els dos clients hi ha el programa però la base de dades està situada en el client que tingui menys càrrega.

4.2. Disseny d'interfícies

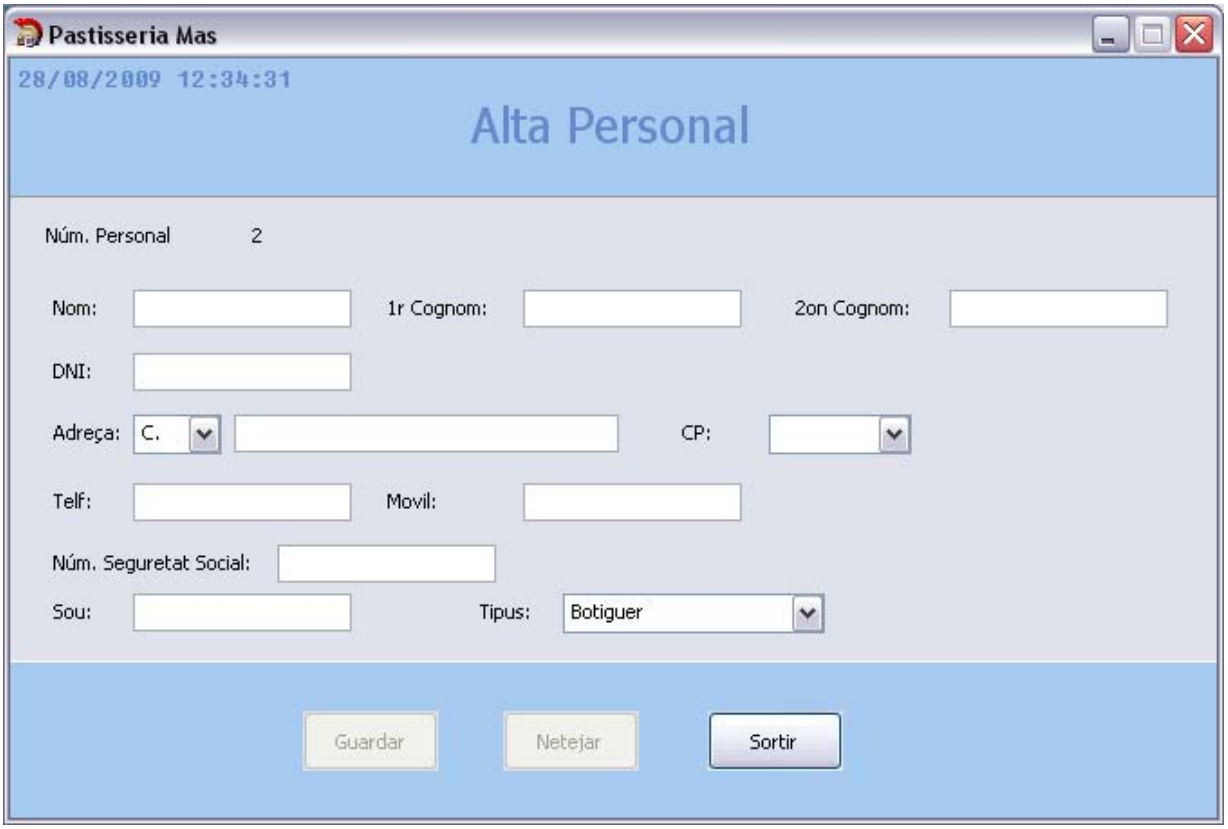
Per tal d'assegurar un bon disseny i mantenir una coherència visual en les interfícies amb l'usuari, s'ha tingut en compte una sèrie de característiques:

- **Mínim esforç:** en totes les finestres, com ja hem comentat en l'apartat d'anàlisis del sistema, es segueix un mateix model perquè l'usuari s'hi pugui adaptar fàcilment. S'ha aconseguit aquesta propietat gràcies a l'herència entre les finestres.
- **Mínima Memòria:** s'han utilitzat els botons justos perquè hi hagi una comunicació senzilla entre l'usuari i el programa .
- **Notificació dels problemes**: si hi ha alguna acció que no pot ser realitzada es mostra per pantalla un missatge sobre com solucionar-ho.
- **Tolerància als errors**: en el cas d'haver-hi un error, permet abandonar l'acció i solucionar-lo.
- **Confirmació en les baixes**: en el cas que algun objecte es vulgui donar de baixa, l'usuari ho ha de confirmar en un missatge que sortirà per pantalla.

Pel què fa a l'entrada de dades, tindran el següent comportament:

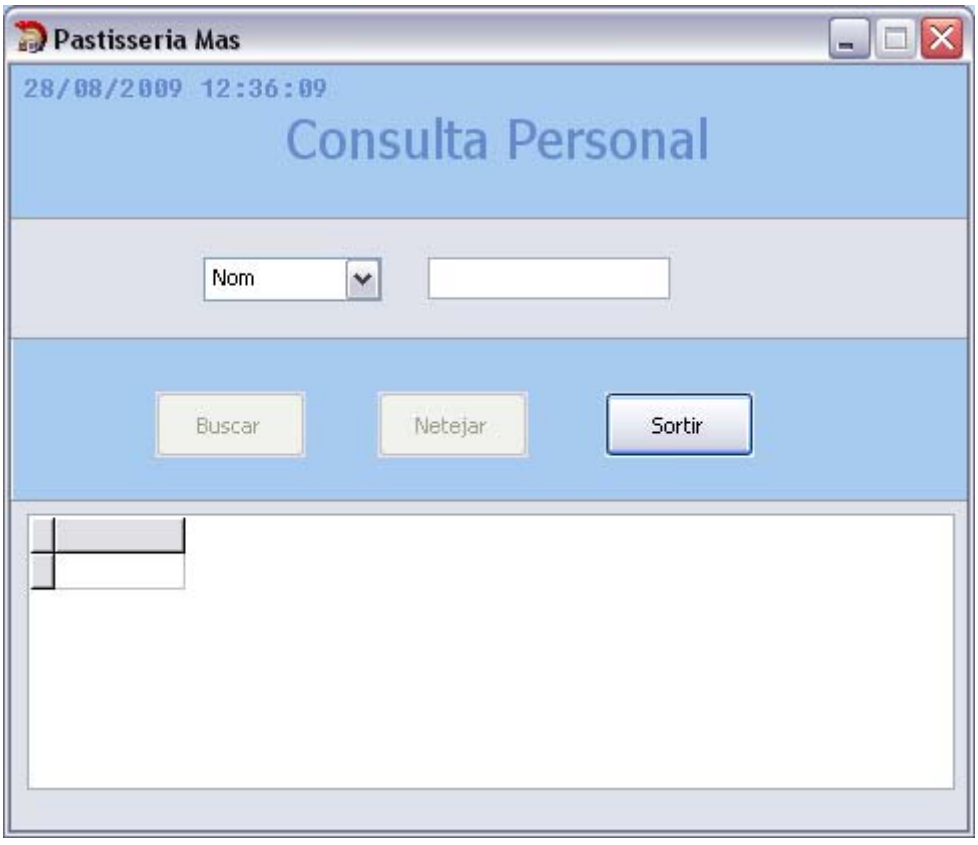
- Quan les condicions necessàries per poder dur a terme les accions no s'hagin complert, els botons estaran deshabilitats.
- En el cas que l'usuari vulgui donar d'alta un registre nou i algun atribut d'aquest estigui relacionat amb una dada existent a la base de dades, apareixerà un *ComboBox* que contindrà totes les possibles opcions.

Seguidament, es mostraran algunes d'interfícies per exemplificar les característiques esmentades anteriorment:

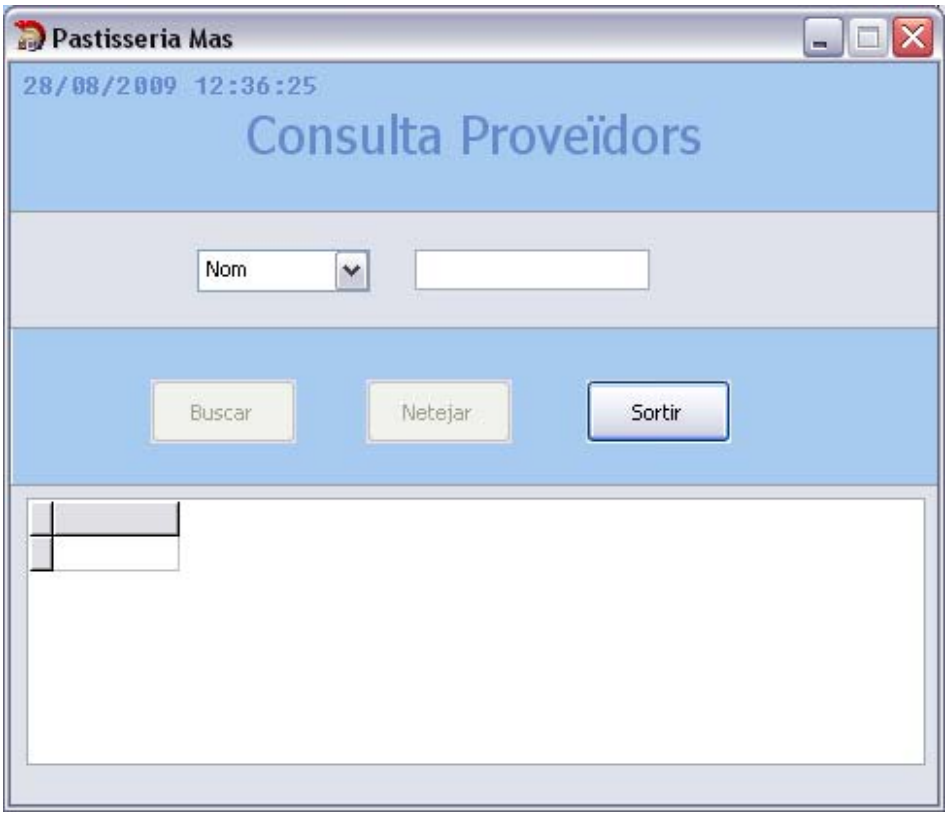


Interfície alta personal

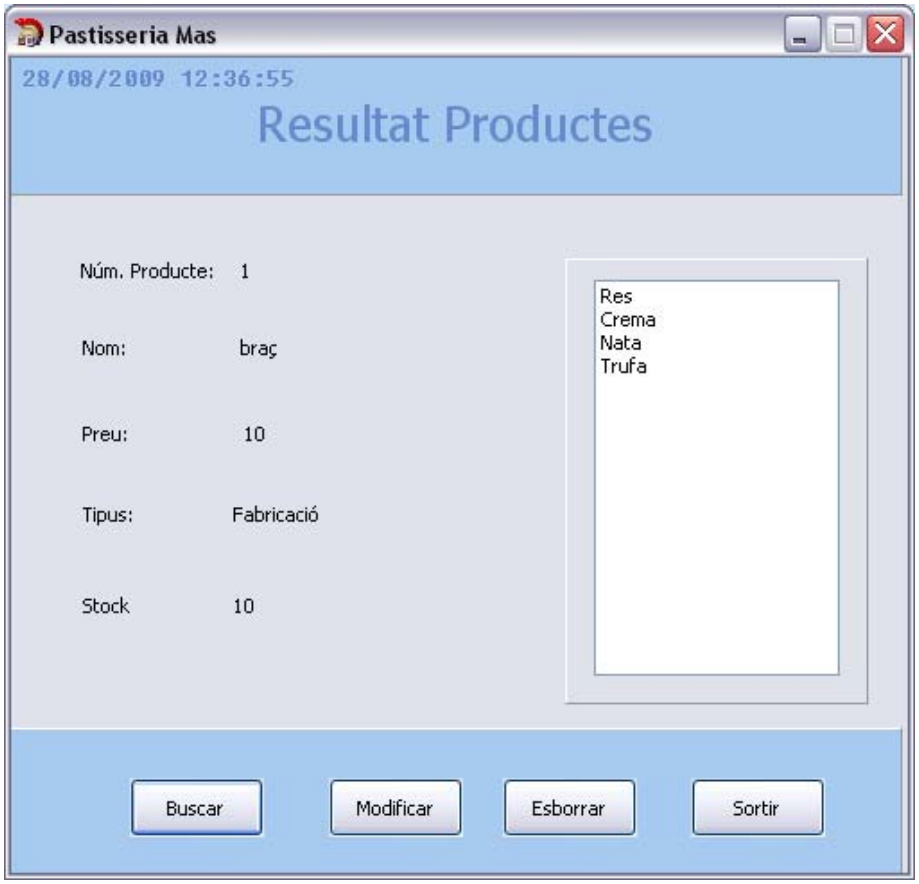
Interfície alta producte



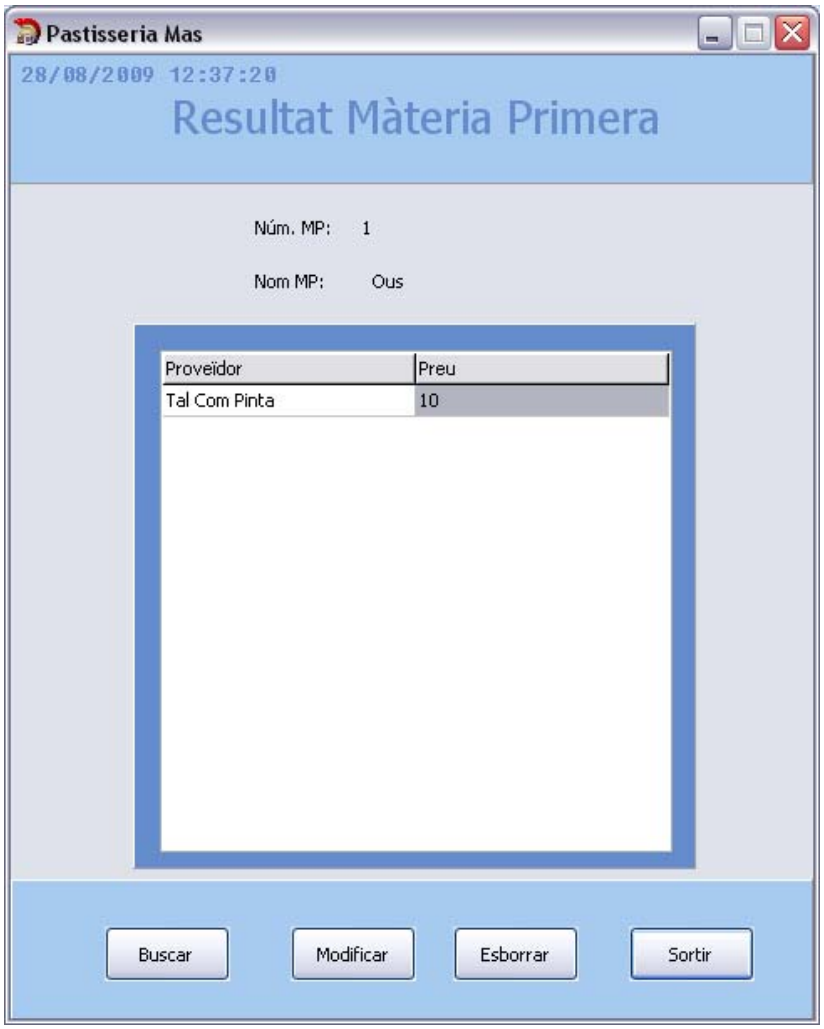
Interfície consulta personal



Interfície consulta proveïdors



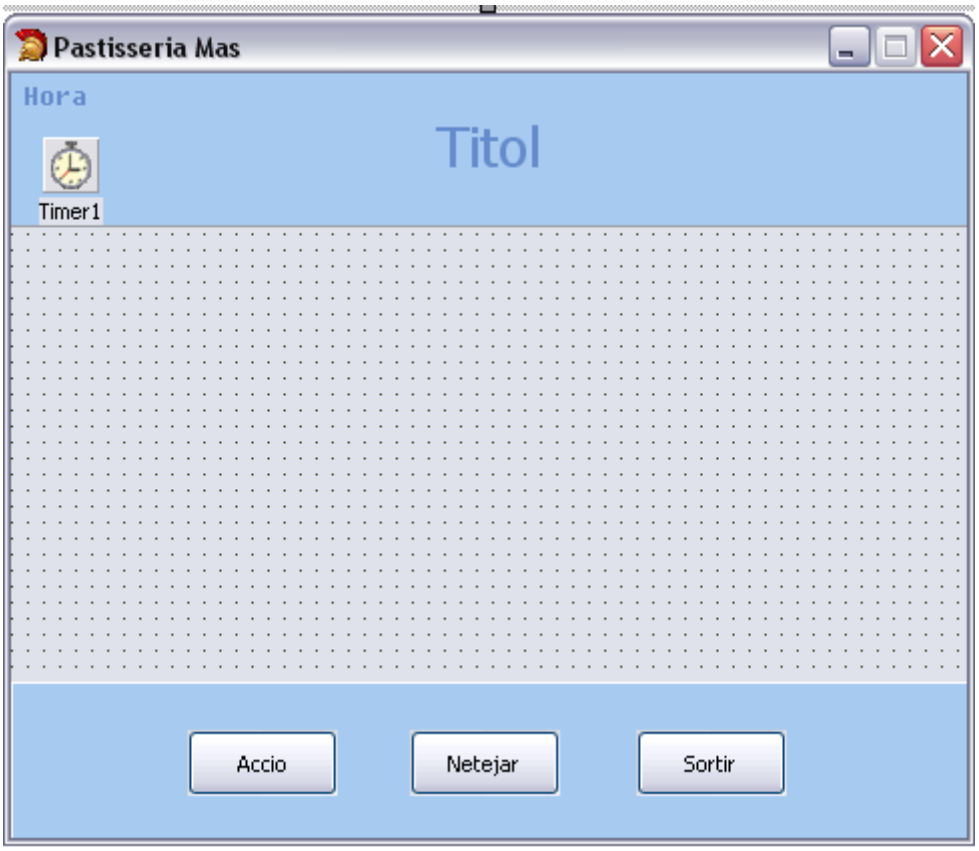
Interfície resultat productes



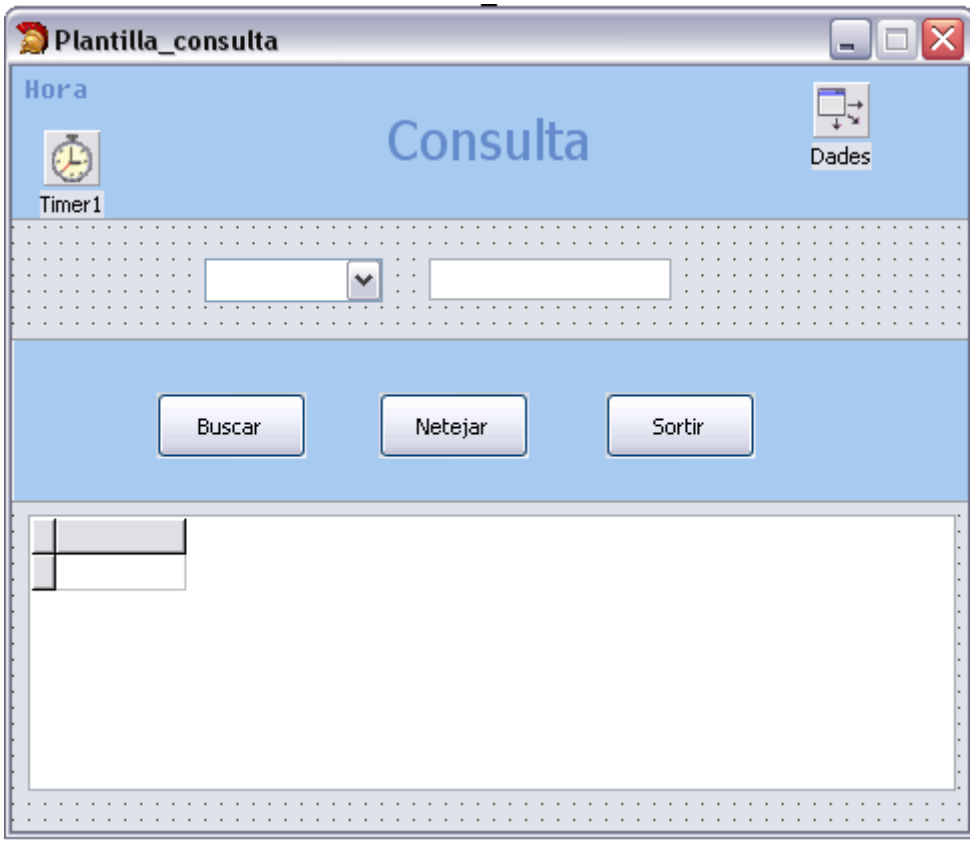
Interfície resultat matèria primera

En tots aquests casos es pot observar que les finestres segueixen el mateix esquema: tenen els mateixos títols, el mateixos botons i la mateixa distribució. S'ha aconseguit aquest esquema utilitzant classes plantilla de manera que totes les finestres han heretat aquestes propietats.

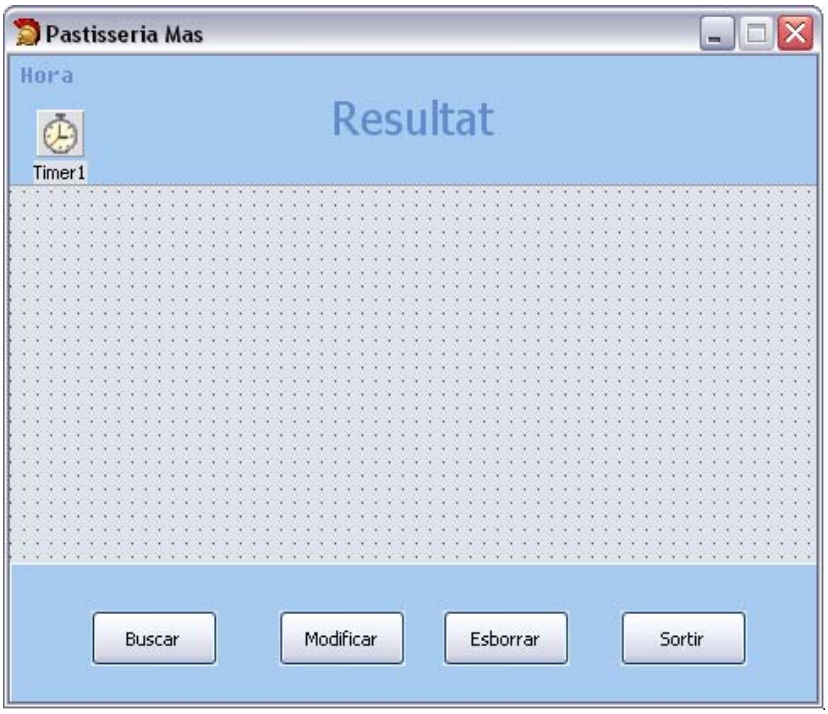
A continuació, es mostraran les plantilles que s'han utilitzat en l'aplicació. La relació d'herència entre les finestres està explicada en l'anàlisi del sistema.



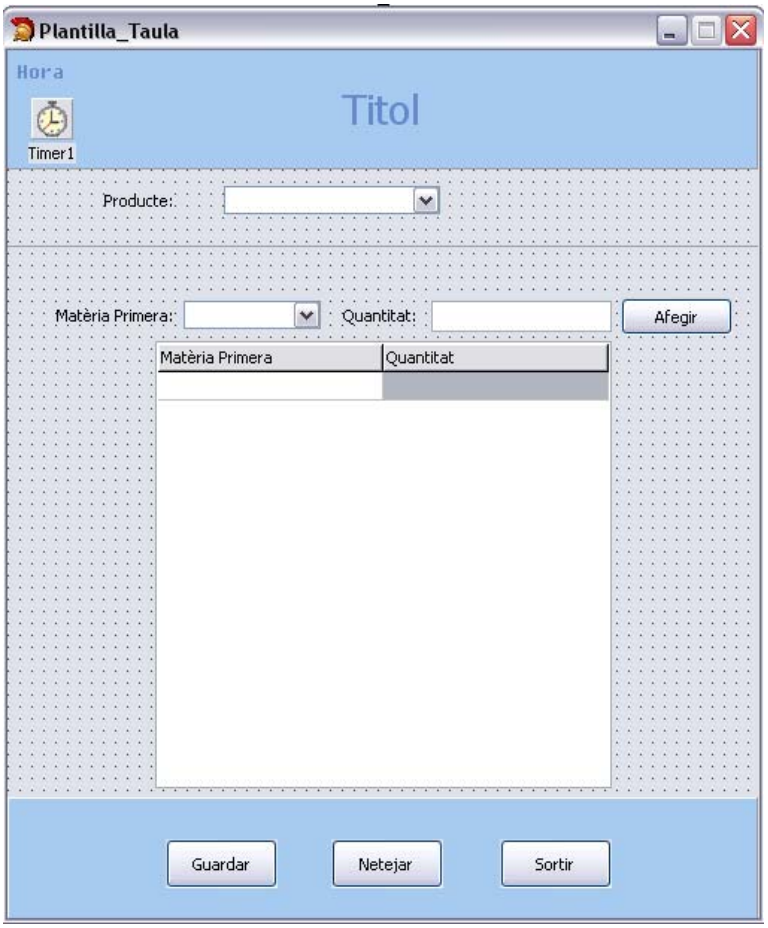
Plantilla finestra



Plantilla finestra consulta

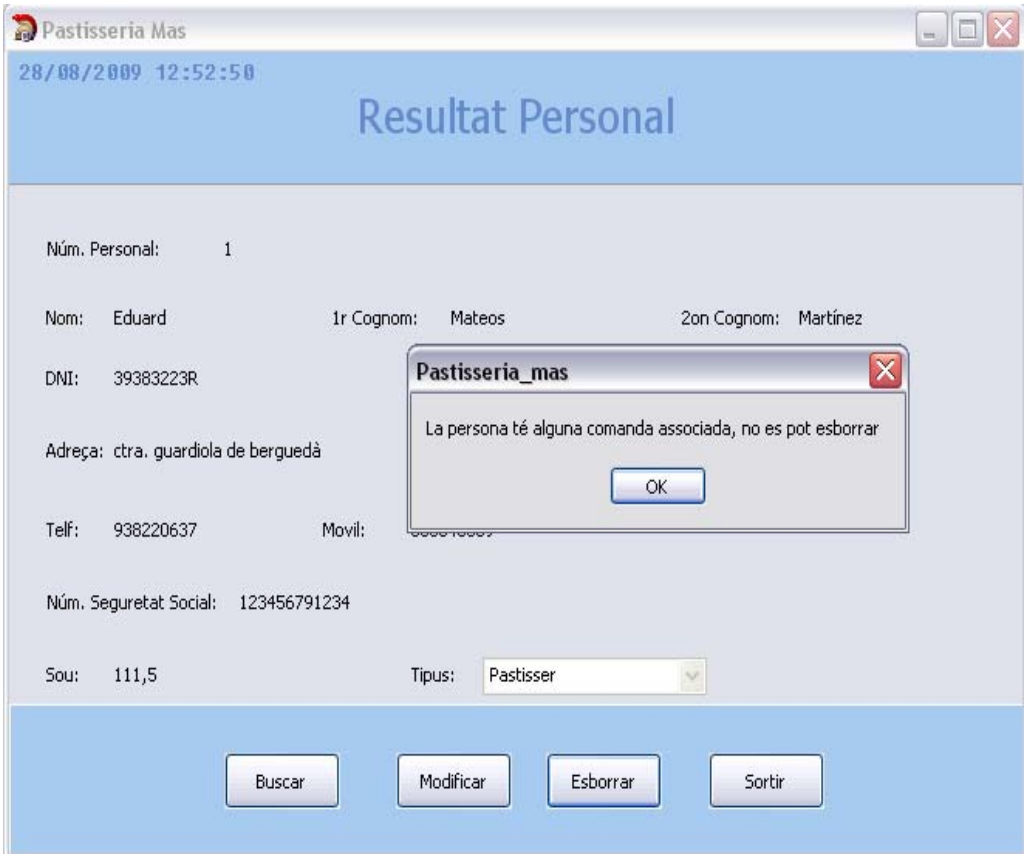


Plantilla finestra resultat cerca



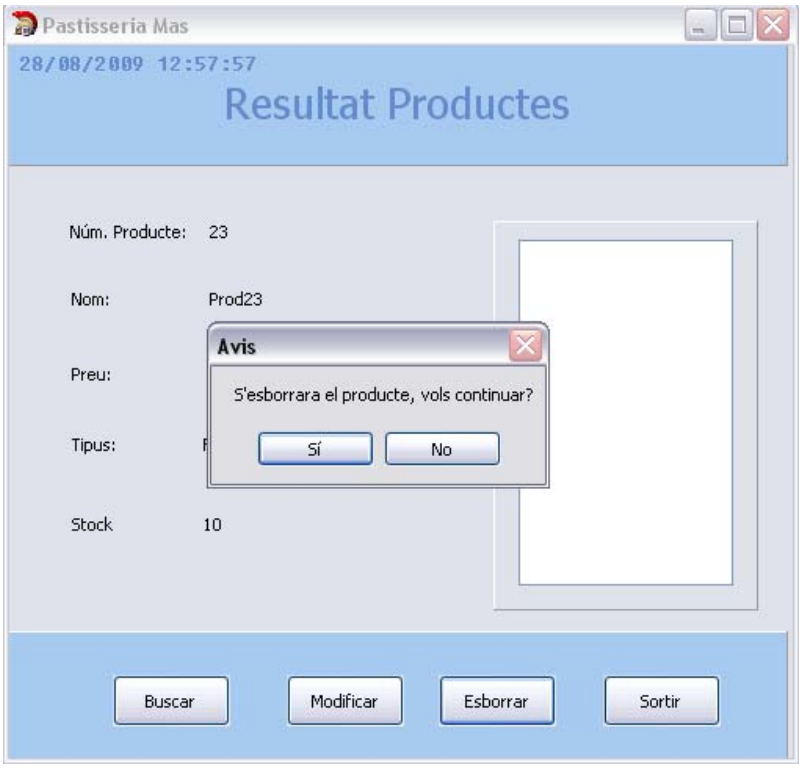
Plantilla finestra taula

A continuació, es mostrarà un exemple del missatge informant dels problemes que poden sorgir en el moment de donar de baixa una persona:



Missatge informant que hi ha un error en donar de baixa una persona

Pel què fa al tema de la baixa, també s'ha comentat que es sol·licita una confirmació per realitzar l'acció. A continuació es mostrà un exemple:



Missatge de confirmació per donar de baixa

Una altre de les característiques que hem comentat és que el botons només estan actius quan es compleixin les condicions necessàries. A continuació es mostra un exemple on es pot veure aquesta propietat:

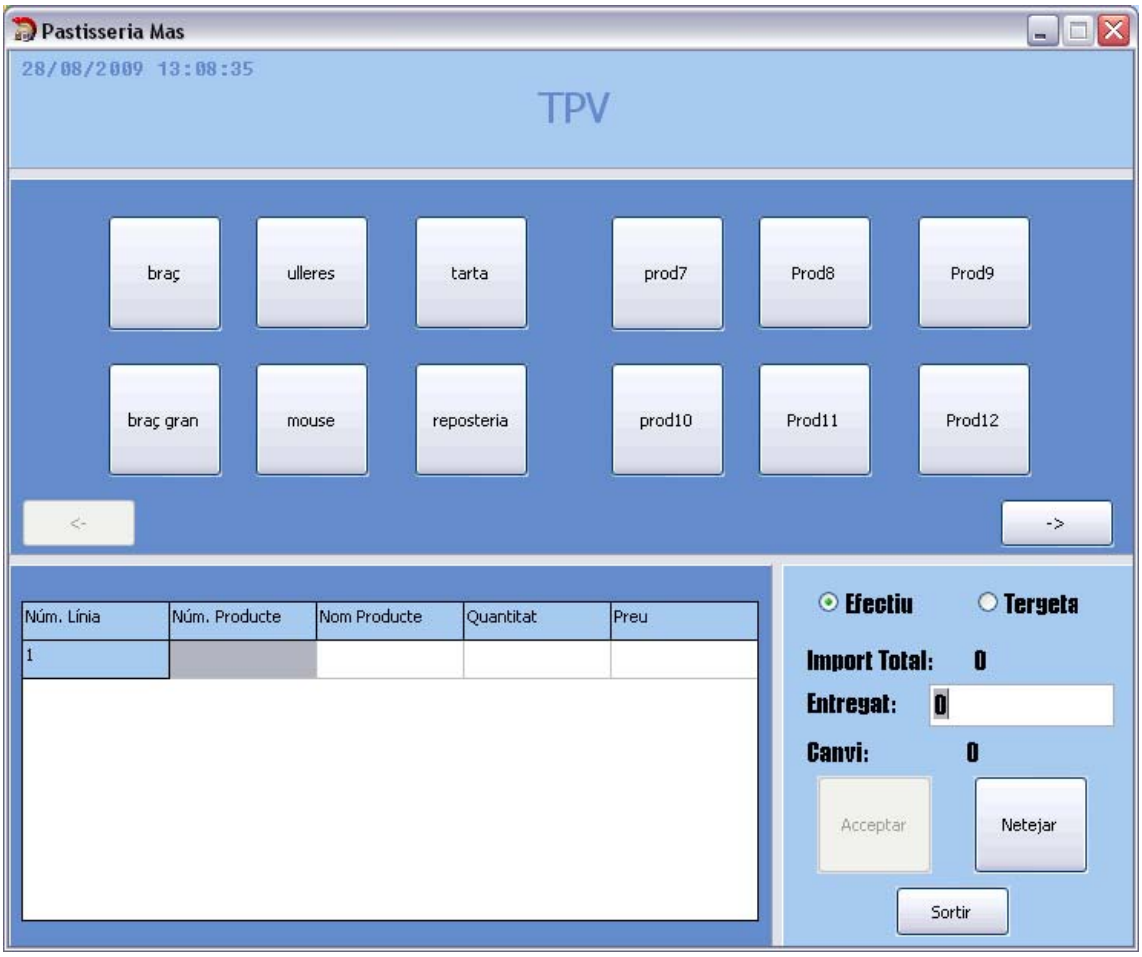
Pantalla amb els botons desactivats

En aquest exemple, podem observar que tan el botó *guardar* com *netejar* estan inactius. En el moment en què s'ha omplert tota la informació necessària, els botons s'activen.

Pantalla amb els botons activats

A més, en aquesta última pantalla també podem veure un *ComboBox* amb les opcions possibles fruit d'una relació a la base de dades.

A continuació, per exemplificar la facilitat i la claredat per l'usuari, es mostrarà la finestra més complexa que és la del TPV:



Pantalla del TPV

4.3. Disseny de la base de dades

Es treballarà amb una base de dades SQL, concretament amb el sistema gestor de base de dades MySQL.

Com que la base de dades no és orientada a objectes, es necessita passar tots els diagrames de classes vistos a l'anàlisi a **entitat-relació**. A continuació es mostrarà el diagrama corresponent:

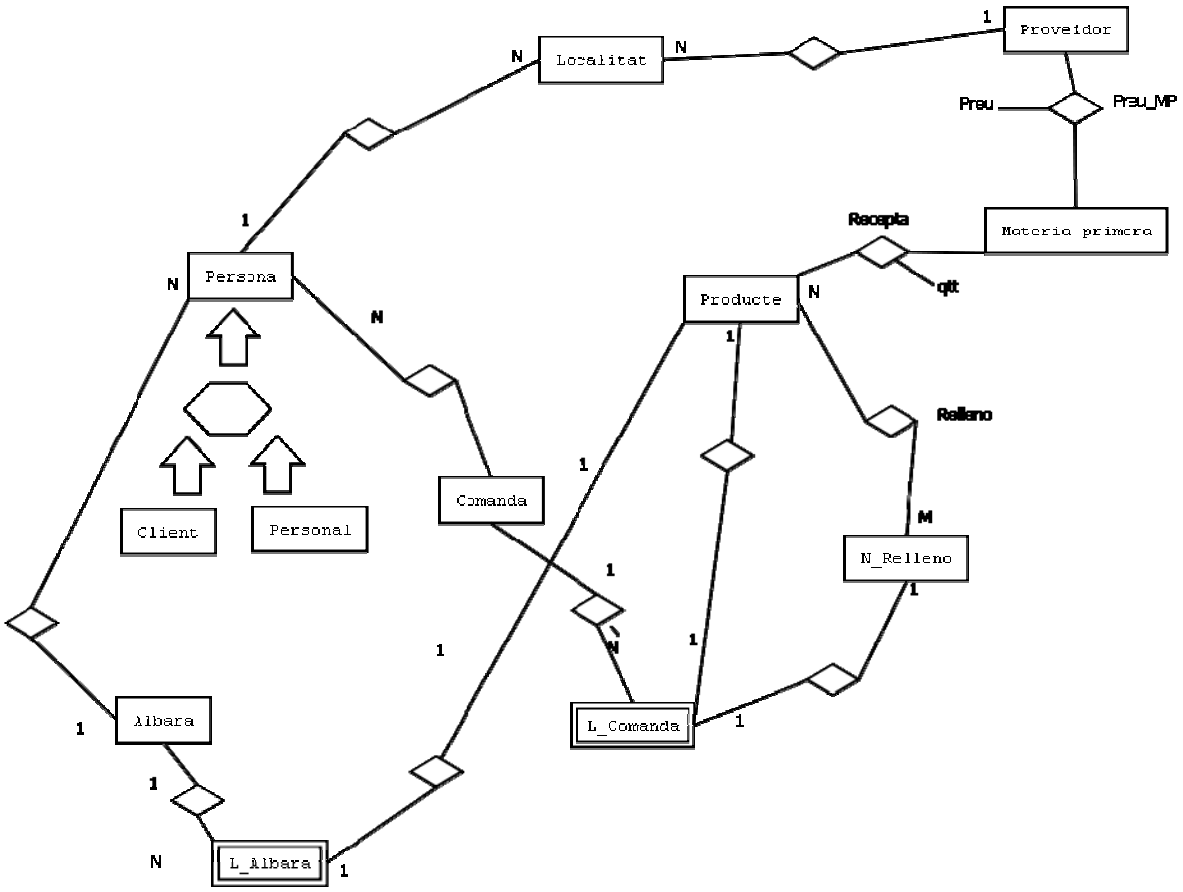


Diagrama entitat-relació

Seguidament, s'exemplificarà la traducció de l'entitat, la comanda, la persona i localitat a nivell de taula:

Comanda (**id**, id_persona, tipus_persona, data, estat);
Persona (**id, tipus_persona**, nom, 1_cognom, 2_cognom, CP, telf, mov)
Localitat (**CP**, Nom)
La negreta corresponent a la clau primària i el subratllat a la clau forana.

A continuació es mostraran les comandes utilitzades per crear tota la base de dades:

```
CREATE DATABASE pastisseria_mas
USE pastisseria_mas
CREATE TABLE materia_primera(
        id SMALLINT PRIMARY KEY,
```

```
        nom CHAR(15));
CREATE TABLE proveidor(
        id SMALLINT PRIMARY KEY,
        nom CHAR(20),
        telf CHAR(10),
        poblacio CHAR(25),
        adreca CHAR(30));
CREATE TABLE preu_mp(
        id_prov SMALLINT NOT NULL,
        id_mp SMALLINT NOT NULL,
        preu float,
        PRIMARY KEY(id_prov,id_mp),
        FOREIGN KEY (id_prov)
        REFERENCES proveidor(id),
        FOREIGN KEY (id_mp)
        REFERENCES materia_primera(id));
CREATE TABLE producte(
        id SMALLINT PRIMARY KEY,
        nom CHAR(15),
        preu float,
        tipus CHAR(1),
        stock SMALLINT);
CREATE TABLE trelleno(
        id SMALLINT NOT NULL PRIMARY KEY,
        nom char(15) NOT NULL);
CREATE TABLE relleno(
        id_prodSMALLINT NOT NULL,
        id_rell SMALLINT NOT NULL,
        PRIMARY KEY (id_prodl,id_rell),
        FOREIGN KEY (id_prod) REFERENCES producte(id),
        FOREIGN KEY(id_rell) REFERENCES trelleno(id));
CREATE TABLE recepta(
        id_prod SMALLINT NOT NULL,
        id_mp SMALLINT NOT NULL,
```

```
        qtt LONG,
        PRIMARY KEY(id_prod,id_mp),
        FOREIGN KEY recpeta_prod(id_prod)
        REFERENCES producte(id),
        FOREIGN KEY  recepta_mp(id_mp)
        REFERENCES  materia_primera(id));
CREATE TABLE localitat(
CP CHAR(5)PRIMARY KEY,
Nom char(25));
CREATE TABLE persona(
        id SMALLINT NOT NULL,
        tipus_persona CHAR(1) NOT NULL,
        nom CHAR(10),
        1_cognom CHAR(10),
        2_cognom CHAR(10),
        CP CHAR(5),
        telf CHAR(10),
        mov CHAR(10),
        PRIMARY KEY(id, tipus_persona),
        FOREIGN KEY (CP)
        REFERENCES localitat(CP));
CREATE TABLE personal(
id SMALLINT NOT NULL,
        tipus_persona CHAR(1) NOT NULL,
        DNI CHAR(9),
        NSS CHAR(12),
        sou FLOAT,
        adreca CHAR (30),
        tipus_personal CHAR(1),
        PRIMARY KEY(id,tipus_persona),
        FOREIGN KEY(id,tipus_persona)
        REFERENCES persona(id,tipus_persona));
CREATE TABLE comanda(
        id SMALLINT NOT NULL PRIMARY KEY,
```

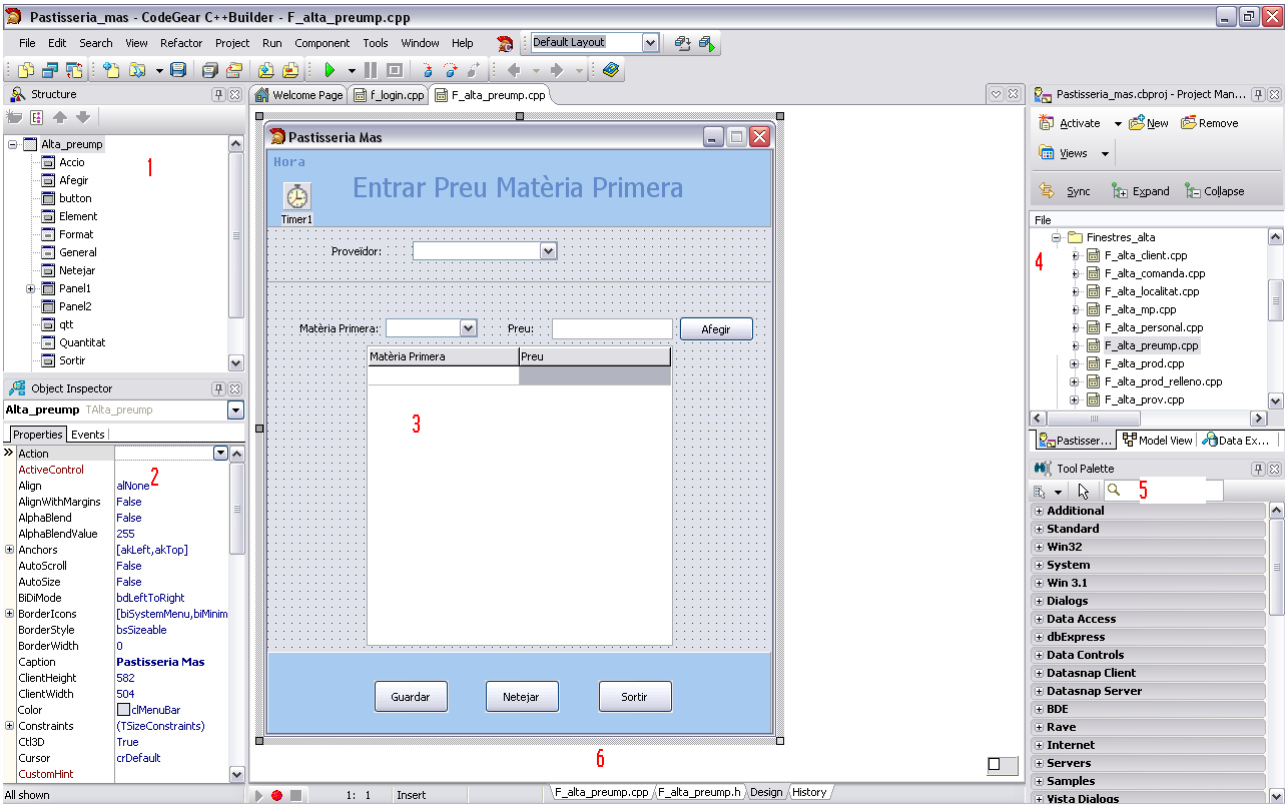
```
    id_persona SMALLINT NOT NULL,
    tipus_persona CHAR(1) NOT NULL,
    dia DATE,
    estat   CHAR(9),
    FOREIGN KEY (id_persona,tipus_persona)
    REFERENCES persona(id,tipus_persona)
    );
CREATE TABLE lcom(
    id_com SMALLINT NOT NULL,
    id_lin SMALLINT NOT NULL,
    id_prod SMALLINT NOT NULL,
    id_rell SMALLINT,
    qtt FLOAT NOT NULL,
    PRIMARY KEY (id_com,id_lin),
    FOREIGN KEY (id_com)
    REFERENCES comanda(id),
    FOREIGN KEY (id_prod)
    REFERENCES producte(id),
    FOREIGN KEY (id_rell)
    REFERENCES trelleno(id));

CREATE TABLE albara(
    id SMALLINT NOT NULL PRIMARY KEY,
    id_persona SMALLINT NOT NULL,
    tipus_persona CHAR(1) NOT NULL,
    dia DATE,
    FOREIGN KEY (id_persona,tipus_persona)
    REFERENCES persona(id,tipus_persona)
    );
CREATE TABLE lalbara(
    id_al SMALLINT NOT NULL,
    id_lin SMALLINT NOT NULL,
    id_prod SMALLINT NOT NULL,
    qtt FLOAT NOT NULL,
```

```
preu Float NOT NULL,
PRIMARY KEY (id_al,id_lin),
FOREIGN KEY (id_al)
REFERENCES albara(id),
FOREIGN KEY (id_prod)
REFERENCES producte(id));
```

4.4. Disseny del programa

El llenguatge utilitzat per implementar aquesta aplicació és el llenguatge C++. Concretament, s'ha escollit el compilador CodeGear™ C++Builder® 2009. A continuació es mostrarà l'entorn de treball mitjançant impressions de pantalla.

El primer apartat de l'aplicació és el *Design* que serveix per fer la interfície:

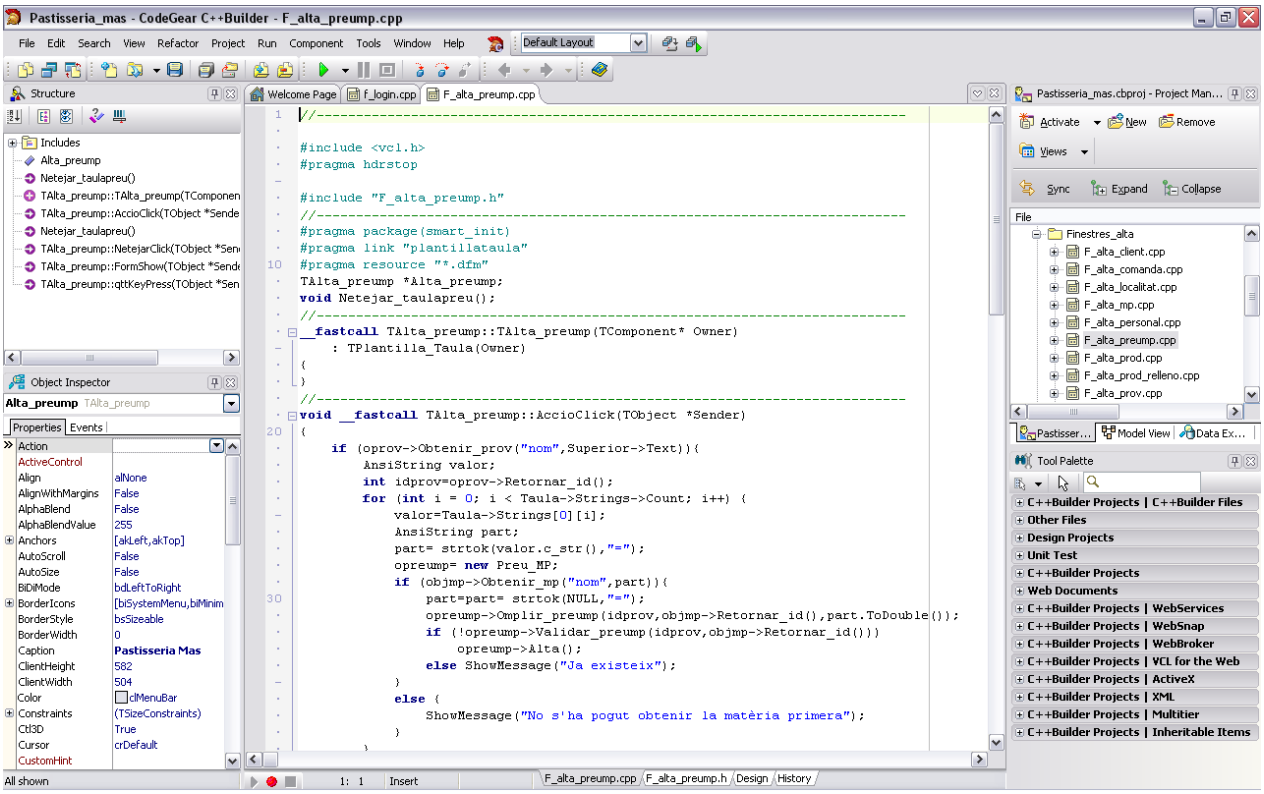


Entorn de treball *CodeGear C++Builder*

En aquesta imatge es pot veure l'entorn de treball del compilador, concretament la part de disseny gràfic. Es pot observar que la part de la pantalla marcada amb un **1** indica els diferents objectes que es troben en aquesta finestra. La zona marcada amb un **2** correspon a les propietats de l'objecte. Quan es selecciona un objecte de

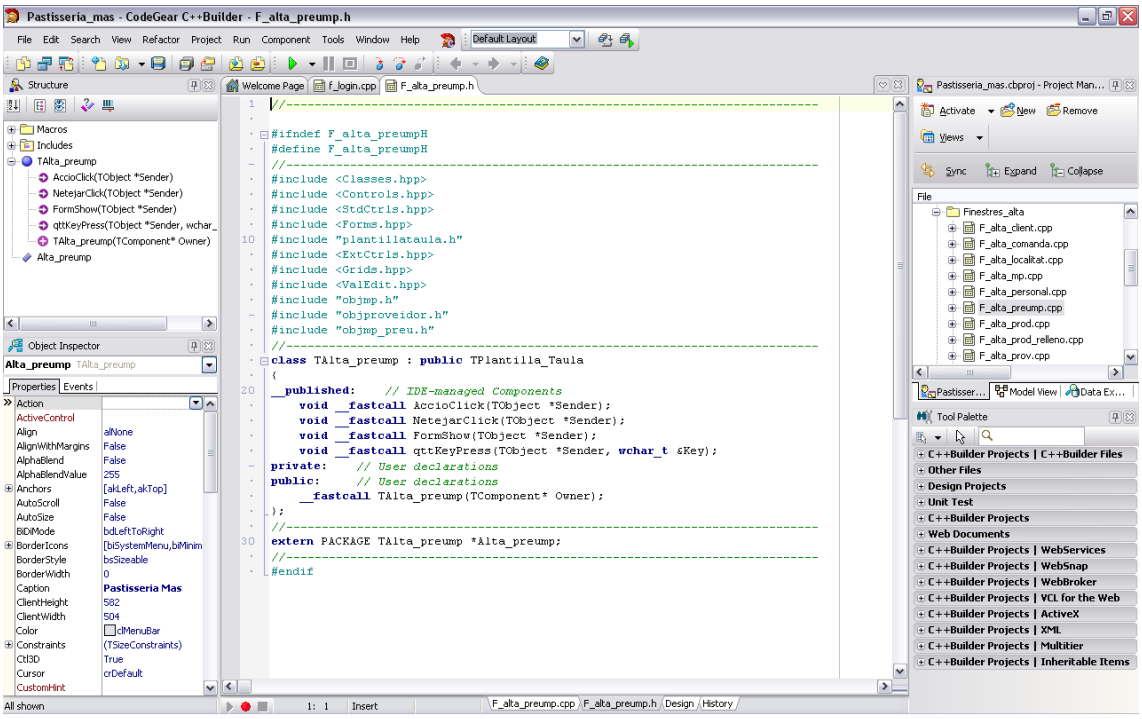
la finestra, en aquesta zona hi figuren totes les seves propietats com el nom, mides, colors... Pel què fa a la zona marcada amb un **3**, correspon a com quedaran col·locats els diferents components que es necessitin. La zona marcada amb un **4** ens indica tots el fitxers existents d'aquell projecte. Pel què fa a la zona **5** correspon a tots els components que es poden utilitzar en el nostre projecte, des de botons, *ComboBox*, accés a BD... Per acabar, el valor marcat amb un **6** serveix per desplaçar-se pels fitxer en què estem treballant.

En primer lloc es mostrarà l'arxiu *.cpp* on trobem el codi de l'aplicació*:*



Entorn de treball de l'arxiu .cpp

A la pròxima pestanya hi figura l'arxiu *.h:*



Entorn de treball de l'arxiu .h

Aquest arxiu s'utilitza per la declaració de components i totes les inclusions necessàries.

Per poder implementar aquesta aplicació són necessaris els objectes. A continuació es mostrarà un exemple de com s'implementa un objecte en el llenguatge C++.

- **Exemple d'objecte:**

L'objecte es dividirà en dos arxius diferents, per una banda tenim l'arxiu *.h* que conté totes les declaracions de l'objecte i l'arxiu *.cpp* que conté el propi codi.

A continuació hi ha un exemple de la declaració de l'objecte **Personal**:

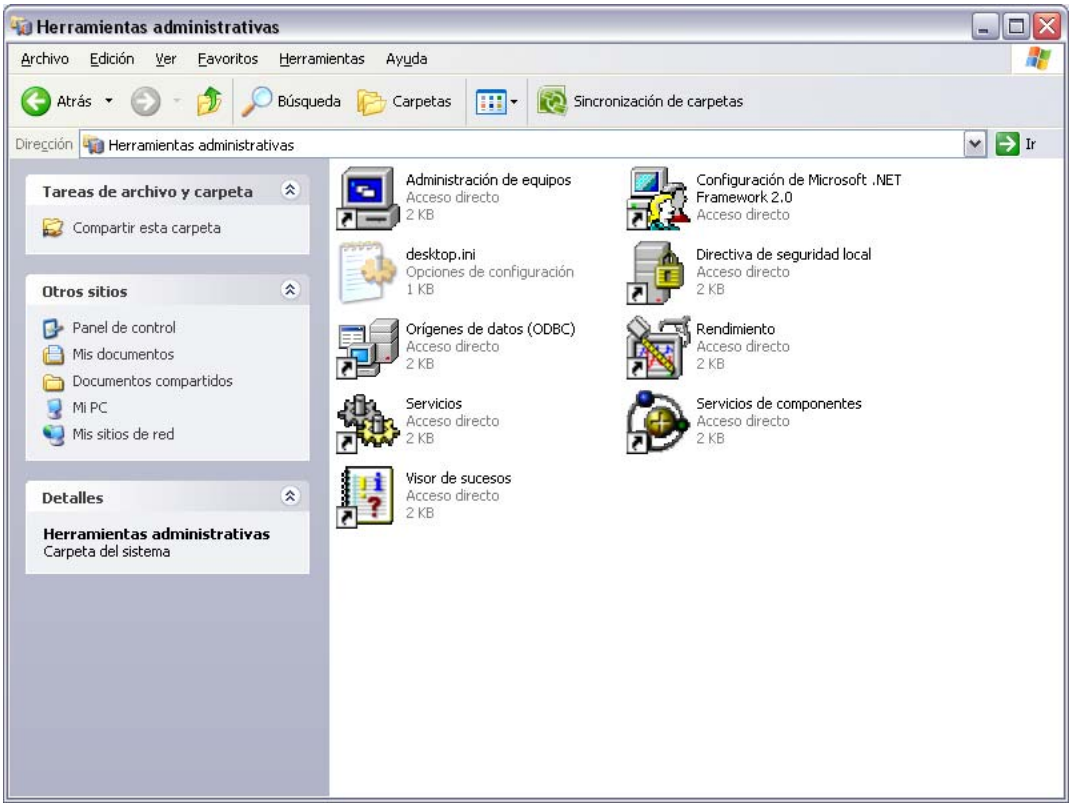
```
#ifndef objpersonalH
#define objpersonalH
#include "objpersona.h"
#include "data.h";
class Personal:public Persona{
        private:
                AnsiString DNI;
```

```
        AnsiString NSS;
        float Sou;
        AnsiString Adreca;
        AnsiString Tipus_personal;
    public:
        void Alta();
        void Baixa();
        void Omplir(int id, AnsiString tipus_persona,AnsiString nom,AnsiString cognom1,
                    AnsiString cognom2,AnsiString cp,AnsiString telf,AnsiString mov,
                    AnsiString dni, AnsiString nss, float sou, AnsiString adreca,AnsiString tipus_personal);
        int Ultim_id();
        bool Validar_personal();
        void Consulta(AnsiString camp, AnsiString valor);
        bool Obtenir_personal(AnsiString id);

        AnsiString Retorna_DNI();
        AnsiString Retorna_NSS();
        AnsiString Retorna_Adreca();
        AnsiString Retorna_tipus_personal();
        float Retorna_sou();
        bool Modificar(Personal *operso);
};
extern PACKAGE Personal *opersonal;


#endif
//---------------------------------------------------------------------------
```

Es pot observar que l'objecte **personal** hereta informació d'un altre objecte, en aquest cas de l'objecte *Persona*. S'han declarat els atributs en privat, és a dir, que només hi pot accedir el propi objecte. Els serveis s'han declarat públics perquè es pugin utilitzar al llarg de tota l'aplicació

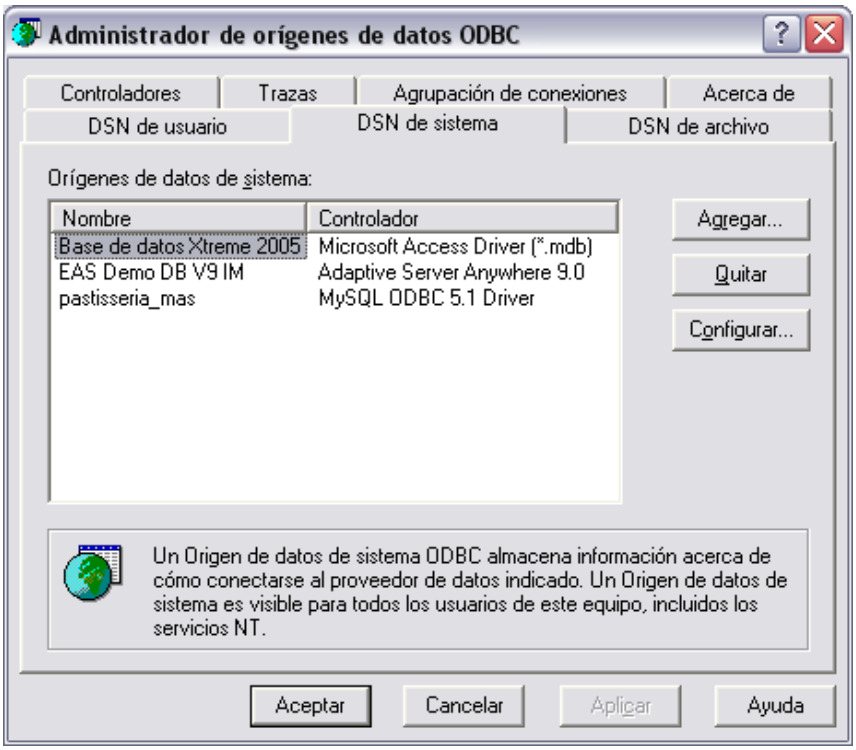
- **Exemple d'accés a la Base de dades**

En aquesta aplicació s'ha d'accedir a una base de dades. Com ja s'ha comentat anteriorment, s'ha realitzat la BD amb *MySQL*. En primer lloc, s'ha de donar accés a aquesta BD creant un origen de dades que servirà per guardar la informació necessària per accedir a la Base de dades. Per fer-ho, amb el *Windows XP* s'ha d'accedir a *MiPC > Panel de control > Herramientas administrativas > Origen de datos(ODBC).*



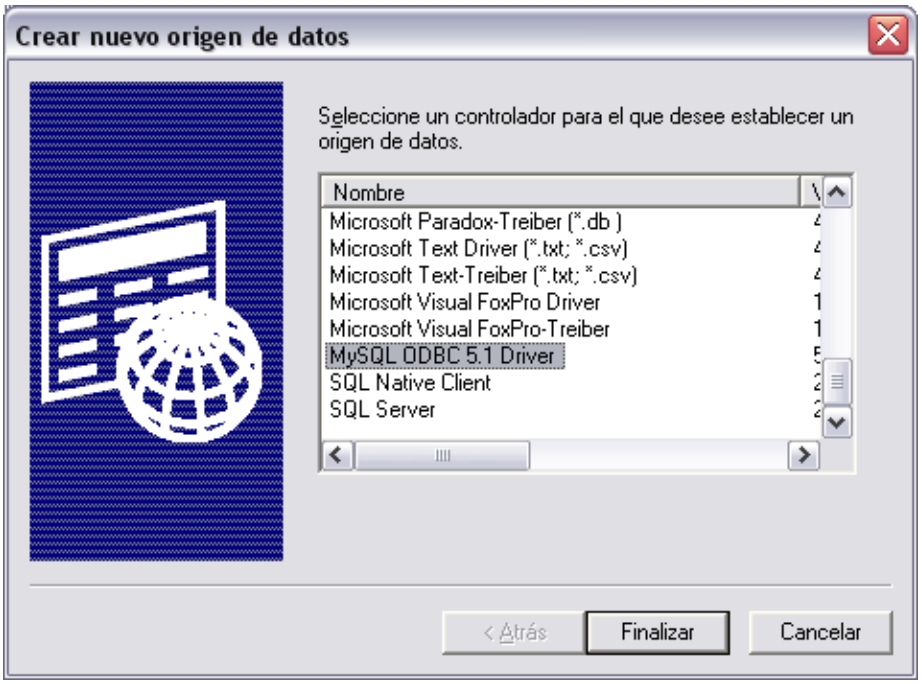
Pantalla Herramientas administrativas de Windows

Un cop hem accedit a aquest accés directe, s'ha d'anar a la pestanya *DNS* del sistema:



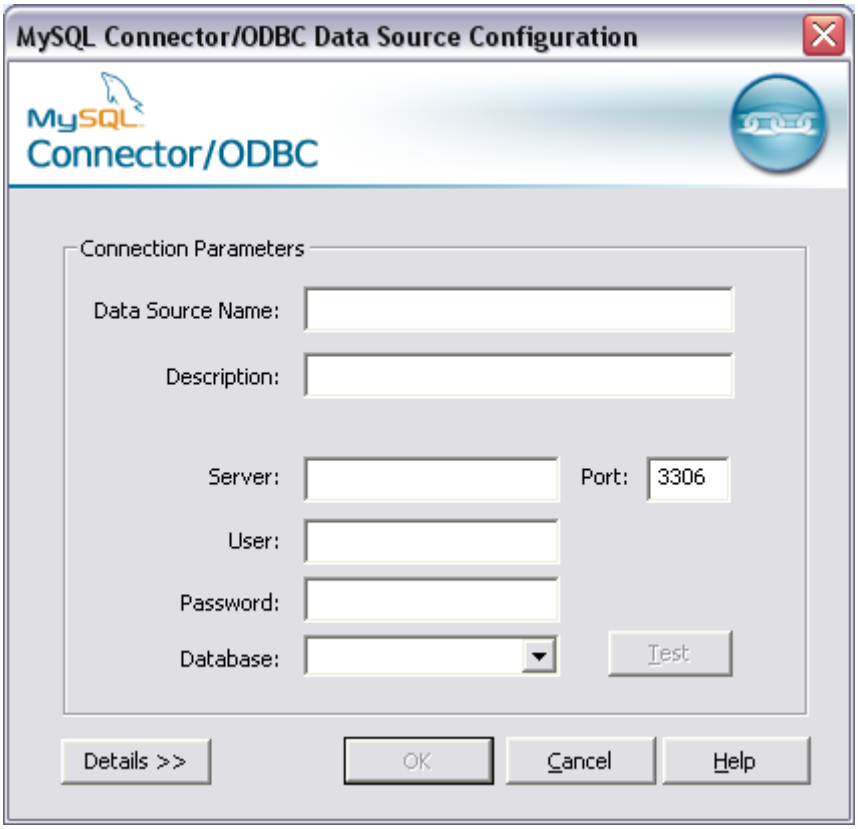
Pantalla DNS del sistema de Windows

Com es pot veure a la imatge, s'ha afegit la informació necessària per accedir a la base de dades de la *pastisseria_mas*. Per poder afegir-ho premem el boto *agregar*. Sortirà un llistat de *drivers* per poder accedir als diferents tipus de bases de dades. En el cas d'aquesta aplicació, el *driver* necessari per accedir és el *MySQL ODBC 5.1*. Aquest no està instal·lat per defecte en el sistema però ens el podem descarregar de la pagina oficial de MySQL.



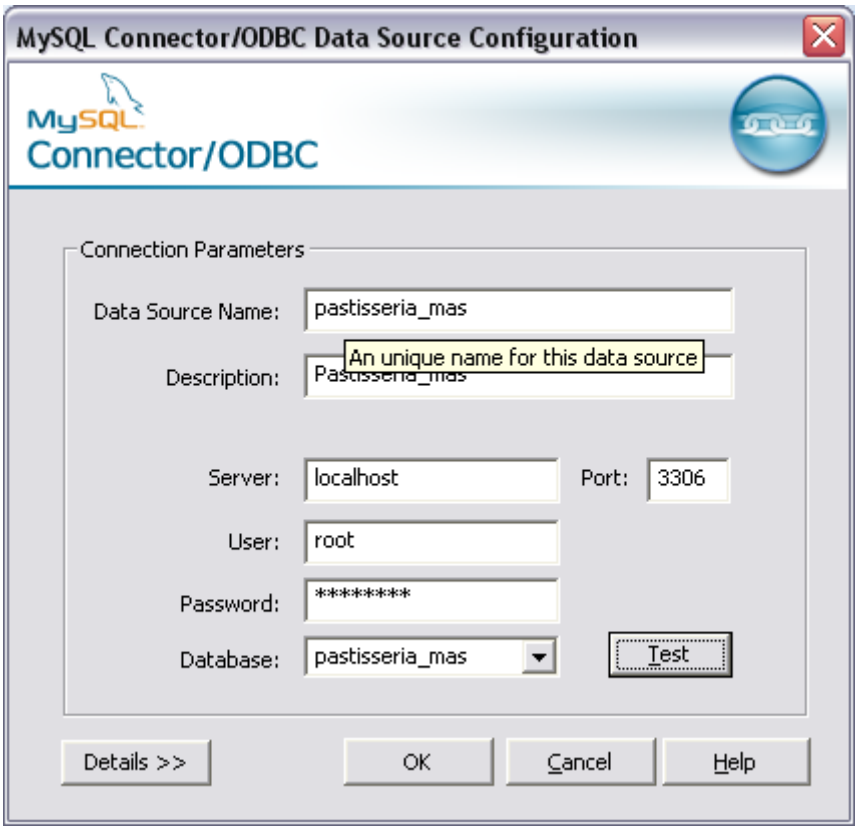
Pantalla per carregar *drivers* de connexió

Un cop seleccionats els *drivers* necessaris de la llista, cal prémer el botó *Finalizar.* Apareixerà la finestra mostrada a continuació:



Pantalla MySQL Connector

En aquesta finestra, s'ha d'omplir les dades corresponents a la nostra base de dades. El camp **Data Source Name** serveix per donar nom a la relació que volem fer. En el següent camp, **Description,** com el seu nom indica, serveix per descriure la base de dades. **Server** és l'adreça IP del sistema que la conté. **User** és el nom necessari per poder accedir-hi. El següent camp, **Password,** serveix per acreditar l'usuari. Finalment, en el desplegable s'ha de seleccionar la base de des amb què volem treballar.



Pantalla MySQL Connector

Cal prémer el botó *OK* i a partir d'ara ja hi haurà la relació establerta.
Un cop establerta la connexió en l'aplicació, s'hi afegeix un *DataModule*. En primer lloc, s'ha d'arrossegar de la paleta de components la **BDE,** una **TDatabase** i tantes **TTable** com taules tingui la base de dades. Per poder realitzar consultes directament a la base de dades, es necessitarà un **TQuery**. En últim lloc, **TSession** permet introduir les dades per poder accedir a la Base de dades. Un cop tenim tots els components, s'hauran de configurar.

En primer lloc, es parlarà de la configuració de la **TDatabase**. A la propietat **AliasName** s'ha de seleccionar del desplegable la base de dades *pastisseria_mas*. En la propietat **LoginPrompt** s'ha de posar *false*, la seva utilitat és que quan s'accedeixi a la BD no demani les dades.

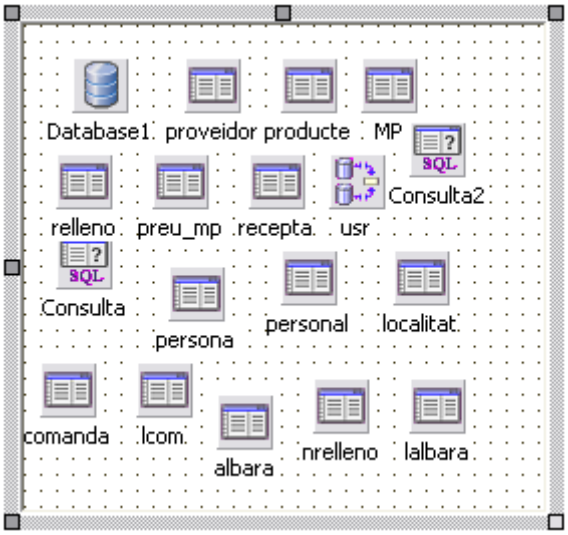
El següent component a configurar serà el **TTable** en la **DatabaseName,** s'hi posarà el nom del component anterior. Seguidament, en la propietat **TableName,** el nom de la taula de la base de dades.

Per configurar el component **TQuery** només s'ha de posar el nom de la base de dades al camp **DatabaseName**.

En últim lloc, es configurarà el component **TSession.** En el camp **SessionName** s'hi posarà el nom de l'usuari per accedir-hi directament. En l'esdeveniment *OnPassword* s'hi escriurà el següent codi:

```
usr->AddPassword("Password");
Continue=true;
```

Aquestes dues línies de codi fan que cada cop que s'intenti accedir a la base de dades s'introdueixi la contrasenya automàticament.
Un cop configurats tots aquests components ja es podran utilitzar per treballar en la base de dades.
El *DataModule* quedarà de la següent manera:



DataModule

Les accions principals que s'han de fer a la Base de dades són l'**alta, la baixa, la consulta i la modificació**. A continuació es mostrarà un exemple de cada funció:

- **Exemple d'Alta**

```
void Persona::Alta(){
    Global::Alta(DataModule1->persona);
    DataModule1->persona->FieldByName("id")->AsInteger=opersona->Id;
```

```
        DataModule1->persona->FieldByName("tipus_persona")-
>AsString=opersona->Tipus_persona;
        DataModule1->persona->FieldByName("nom")->AsString=opersona->Nom;
        DataModule1->persona->FieldByName("1_cognom")->AsString=opersona-
>Cognom1;
        DataModule1->persona->FieldByName("2_cognom")->AsString=opersona-
>Cognom2;
        DataModule1->persona->FieldByName("CP")->AsString=opersona->CP;
        DataModule1->persona->FieldByName("telf")->AsString=opersona->Telf;
        DataModule1->persona->FieldByName("mov")->AsString=opersona->Mov;
        DataModule1->persona->Post();
        DataModule1->persona->Close();
};
```

Pel què fa el Global::Alta(DataModule->Persona); el codi és el següent:

```
        virtual void Global::Alta(TTable *ta){
                ta->Open();
                ta->Last();
                ta->Insert();
        };
```

A continuació s'explicarà una mica el codi de l'alta:

En primer lloc, la funció *Global::Alta* obre la taula per poder inserir-hi informació i activar el mode *inserir*. Un cop ja s'ha adaptat la taula, es procedeix a la inserció de dades. S'utilitzarà el següent servei: **FieldByName("id")->AsInteger**.

La seva funció és accedir a un camp concret de la taula, en aquest cas el camp *id.* A més, s'indica el tipus de camp que es tracta i es fa una assignació del valor que es vol registrar.

DataModule1->persona->FieldByName("id")->AsInteger=opersona->Id;

Un cop realitzades totes aquestes insercions, es donen les ordres:

DataModule1->persona->Post(); Serveix per obligar a la base de dades a memoritzar-ho.
DataModule1->persona->Close(); Serveix per tancar la taula un cop ja s'ha fet la inserció.

- **Exemple de Baixa**

```
void Producte::Baixa(){
        Global::Baixa("DELETE FROM producte WHERE id="+(AnsiString)Id+";");
};
```

Pel què fa la funció *Global::Baixa()* el codi és el següent:

```
        virtual void Global::Baixa(AnsiString com){
                DataModule1->Consulta->Active=false;
                DataModule1->Consulta->Close();
                DataModule1->Consulta->SQL->Clear();
                DataModule1->Consulta->SQL->Add(com);
                DataModule1->Consulta->ExecSQL();
                DataModule1->Consulta->Close();
        };
```

En aquest cas, el pes de donar de baixa un objecte recau sobretot en la funció global de la baixa. S'ha recorregut a realitzar una baixa com si s'estigués directament a la base de dades. Es passa directament la comanda a la funció mitjançant un **TQuery** que realitzarà l'acció. En aquest cas, la comanda és:

"DELETE FROM producte WHERE id="+(AnsiString)Id+";"

- **Exemple de Consulta**

```
void Proveidor::Consulta(AnsiString camp, AnsiString valor){
                DataModule1->Consulta->Active=false;
                DataModule1->Consulta->Close();
                DataModule1->Consulta->SQL->Clear();
                if (camp=="id") {
```

```
            DataModule1->Consulta->SQL->Add("select id, nom FROM proveidor
where id like '%"+valor+"%'");
            }
            else{
                  DataModule1->Consulta->SQL->Add("select id, nom FROM
proveidor p where nom like '%"+valor+"%'");
            }
            DataModule1->Consulta->Prepare();
            DataModule1->Consulta->Open();
            DataModule1->Consulta->Active=true;
};
```

En aquesta funció també es recorre a utilitzar la **TQuery.** Com es pot veure en l'exemple, se l'hi han de passar dos paràmetres. El primer, anomenat *camp*, ens indica quin és l'atribut que es vol consultar i el paràmetre *valor* ens indica què es vol buscar. Si es busca algun nom a la consulta SQL, en lloc de posar el símbol **=** s'hi posa el **like.** En el nom a consultar es posa un % al davant i al darrera que serveix per fer una consulta parcial.

- **Exemple de Modificació**

```
bool Producte::Modificar_producte(Producte *oprod2){
      if (Nom!=oprod2->Retronar_nom()) {
            Nom=oprod2->Retronar_nom();
            Global::Modificar("UPDATE producte set nom='"+Nom+"' where
id="+Id+";");
      }
      if (Preu!=oprod2->Retornar_preu()) {
            Preu=oprod2->Retornar_preu();
            //////Diferents formats de float entre el c++ i el mysql un vol , i l'altre .
            AnsiString p;
            char* pc;
            p=(AnsiString)oprod2->Retornar_preu();
            pc=p.c_str();
```

```
                int i=0;
                while (i<=p.Length()&& pc[i]!=','){i++;}
                if (i!=p.Length()) {
                        pc[i]='.';
                }
                //////^^funcio que modifica la coma per un punt.
                Global::Modificar("UPDATE producte set preu="+(AnsiString)pc+" where id="+Id+";");
        }
        if (Tipus!=oprod2->Retornar_tipus().SubString(0,1)) {
                Tipus=oprod2->Retornar_tipus().SubString(0,1);
                Global::Modificar("UPDATE producte set tipus='"+Tipus+"' where id="+Id+";");
        }
        if (Stock!=oprod2->Retornar_stock()) {
                Stock=oprod2->Retornar_stock();
                Global::Modificar("UPDATE producte set stock='"+(AnsiString)Stock+"' where id="+Id+";");
        }
        return true;
};
```

Aquesta funció serveix per modificar a la base de dades l'objecte *producte*. En primer lloc, se li ha de passar com a paràmetre un objecte del mateix tipus. Es comparará cada atribut dels dos objectes, en el cas que siguin diferents s'utilitzarà un altre cop el **TQuery** per poder modificar els camps. També s'utilitzarà la funció *Global::Modificar* que és molt semblant a la *Global::Baixa* però, en aquest cas, la comanda serà diferent.

- **Exemple de la funció *obtenir* matèria primera**

De totes les funcions que es treballen en la base de dades, la més utilitzada és la d'obtenir un objecte. A continuació es mostrarà el codi de la funció *obtenir* pertanyent a l'objecte MP.

```
bool MP::Obtenir_mp(AnsiString camp,AnsiString var){

        DataModule1->Consulta->Active=false;
        DataModule1->Consulta->Close();
        DataModule1->Consulta->SQL->Clear();
        if (camp=="id") {
            DataModule1->Consulta->SQL->Add("SELECT * FROM
materia_primera m WHERE m.id="+var);
        }
        else {
            DataModule1->Consulta->SQL->Add("SELECT * FROM
materia_primera m WHERE m.nom='"+var+"'");
    }
        DataModule1->Consulta->Prepare();
        DataModule1->Consulta->Open();
        DataModule1->Consulta->Active=true;
        if (DataModule1->Consulta->IsEmpty()) {
            DataModule1->Consulta->Close();
            DataModule1->Consulta->Active=false;
            return false;
        }
        Id=DataModule1->Consulta->FieldByName("id")->AsInteger;
        Nom=DataModule1->Consulta->FieldByName("nom")->AsString;
        DataModule1->Consulta->Close();
        DataModule1->Consulta->Active=false;
        return true;
};
```

- **Exemple de Calcular la quantitat de matèries primeres**

Continuant amb la base de dades, una de les consultes més complexes es la pertanyent a la utilitat de **calcular la quantitat de matèries primeres** per satisfer unes comandes. La consulta és la següent:

SELECT round(sum(recepta.qtt*lcom.qtt),2), materia_primera.nom
FROMpastisseria_mas.lcom INNER JOIN pastisseria_mas.producte ON (lcom.id_prod = producte.id)INNER JOIN pastisseria_mas.recepta ON (recepta.id_prod = producte.id) INNER JOIN pastisseria_mas.materia_primera ON (recepta.id_mp = materia_primera.id), comanda c
WHERE c.dia BETWEEN '"+inici+"' and '"+fi+"'"+"
GROUP BY materia_primera.id;"

4.5. Disseny Pàgina Web

La pàgina web està desenvolupada amb el llenguatge PHP i es connecta a la mateixa base de dades que el programa. L'objectiu d'aquesta web és accedir a distància i des de qualsevol sistema a la informació sobre les vendes que s'han realitzat en el **TPV.**
Aquesta pàgina només la utilitzarà el propietari, per tant, en lloc de realitzar una gestió de diferents usuaris s'ha optat per utilitzar els arxius .**httaccess** i .**httpasswd** per protegir l'entrada a la web.

Es començarà a parlar del **.httaccess** que conté les següents línies:
php_flag session.bug_compat_42 0
php_flag session.bug_compat_warn 0
AuthType Basic
AuthName "Administration"
AuthUserFile c:\xampp\htdocs\web\.htpasswd
require user admin

Les dues primeres línies serveixen per evitar problemes de sessió que hi ha en determinades versions del llenguatge php.
La línia: *AuthType Basic* ens indica quin tipus d'autentificació s'utilitza, en aquest cas es bàsica. La següent línia serveix per mostrar el lloc on vols accedir, es sol·licitarà l'usuari i la contrasenya.

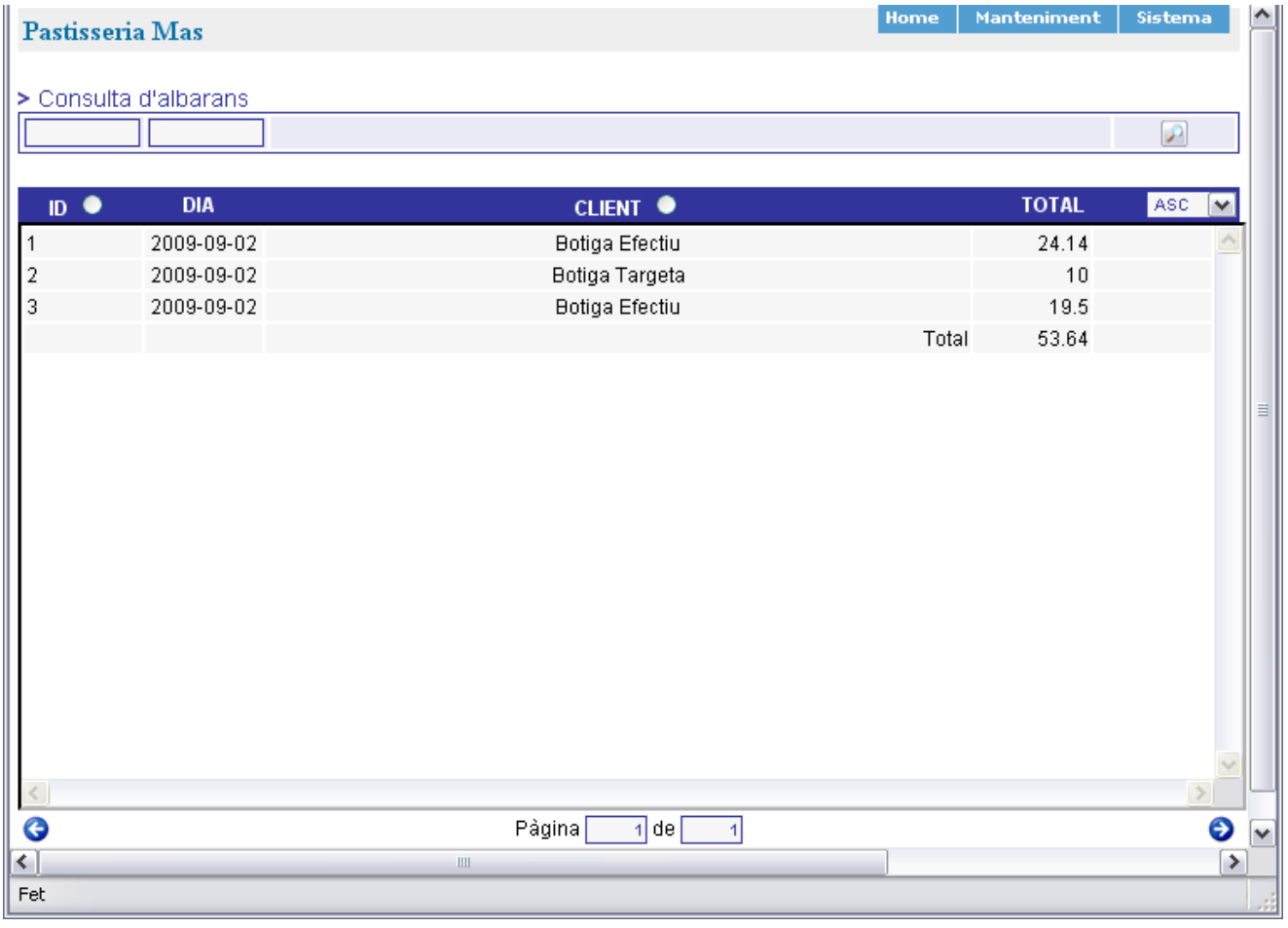
La línia següent serveix per indicar la ruta on es troba l'arxiu **.httpassw.** Normalment, aquest arxiu està en una zona fora de l'accés web. La última línia ens indica quins són els usuaris que poden accedir-hi.

El següent arxiu és el **.httpassw** que conté la relació entre els usuaris i la seva contrasenya, **usr:pass.** En aquest cas només hi apareixerà la següent línia:

admin:$apr1$7hyACVqQ$AoXF96kTEgX/mc8xe2rF3.

La contrasenya està encriptada en MD5, l'original és 1234.
Com ja s'ha comentat anteriorment, aquest arxiu ha d'estar fora de la zona pública de la web.

A continuació, es mostrarà una captura de pantalla corresponent a la pàgina web i s'explicarà com accedeix la web a la base de dades.



Pàgina web de la Pastisseria Mas

La pàgina web permet consultar els albarans donant una data d'inici, una de fi o un interval entre dues dates. També mostra el total de tots els imports dels albarans. Permet ordenar la informació de manera ascendent o descendent segons la data o el client.

Per poder accedir a la base de dades s'ha utilitzat el següent codi:

```
/*---------------- inici variables globals de la classe ---------------*/
	var $database 		= "pastisseria_mas";
	var $mysql_user 	= "root";
	var $mysql_password = "D4nt3061";
	var $mysql_host 	= "localhost";
	var $dbh;
	/*---final variables globals de la classe----------------*/

	/*---------------Contructor de la classe----------*/

	function conexio()
	{
		$this->dbh = mysql_connect( $this->mysql_host, $this->mysql_user, $this->mysql_password);
		mysql_select_db( $this->database, $this->dbh);
		session_start();
	}

	function _conexio()
	{
		mysql_disconnect($this->dbh);
	}
```

Inicialment, es declaren les variables necessàries per poder establir la connexió. Un cop fet, s'utilitza la funció php *mysql_connect* on se li passen els paràmetres declarats anteriorment per poder accedir al SGBD. A continuació, s'accedeix a la base de dades concreta. En últim lloc, s'inicia la comunicació. Per desconnectar-la, s'utilitza la funció *mysql_disconnect.*

Per obtenir un camp concret de la base de dades s'utilitza la següent funció:

```
function getField($_camp, $_taula)
	{
		$query = "SELECT ".$_camp." as ret FROM ".$_taula." ;";

		$result = mysql_query( $query ) OR die(mysql_error().' : '.$query);

		if($myrow = mysql_fetch_array($result))
		{
			return $myrow['ret'];
		} else {
			return "";
		}
	 }
```

5. Manual d'instal·lació del programa

Per realitzar la instal·lació es necessitaran els següents programes:

- mysql-essential-6.0.3-alpha-win32.msi
- mysql-connector-odbc-5.1.5-win32.msi
- bde.exe

En primer lloc, s'haurà d'instal·lar el **SGBD** fent doble clic a l'arxiu *mysql-essential-6.0.3-alpha-win32.msi*. S'han de seguir tots els passos i s'ha de configurar el **SGBD** de manera estàndard. La contrasenya que s'ha d'introduir és ***D4nt3061.***

Seguidament, es procedirà a instal·lar el **connector ODBC**. S'ha de fer doble clic sobre l'arxiu *mysql-connector-odbc-5.1.5-win32.msi* i seguir les instruccions que surten per pantalla.

Després, s'instal·larà la pròpia base de dades ja que es disposa de l'*script* en *SQL* per crear tota l'estructura. S'ha d'accedir a *MySQL Server > MySQL Command Line Client* (dins del menú *d'inici*). A dins la consola s'ha de posar la contrasenya que li hem assignat anteriorment (***D4nt3061***) i escriure la següent comanda: **source ruta_arxiu/pastisseria_mas.sql.**

D'aquesta manera, ja tenim la base de dades creada. A continuació s'instal·larà l'arxiu *bde.exe Borland Database engin* que es necessita pel funcionament correcte del programa.

Finalment, només falta executar l'aplicació. La contrasenya per defecte de qualsevol usuari és 1234.

6. Millores i Conclusions

6.1. Millores

Algunes de les millores que es poden realitzar sobre l'aplicació són les següents:

- Impressió de tiquets:

Degut a problemes de compatibilitat, no s'ha pogut realitzar la impressió de tiquets, ja que el *report* que duu incorporat el compilador té errors interns en la versió actual, que encara no estan corregits. A més, les versions anteriors no són compatibles amb el compilador utilitzat. En el cas que el programa sigui utilitzat per l'empresa, s'hauria d'adquirir la llicència d'algun *report* extern compatible. Les opcions podrien ser tan el *QuickReport* com el *FastReport*.

- Personalització dels usuaris d'accés:

Actualment, només hi ha tres tipus d'usuaris: el pastisser, el propietari i el botiguer. Seria interessant poder donar d'alta als usuaris individualment amb el seu propi *password* i amb privilegis propis segons les tasques a realitzar, permetent modificar-los.

- Gestió de sol·licituds de la botiga:

Quan un client sol·liciti un producte a la botiga, immediatament es donaria un avís al sistema situat a la pastisseria, sense que el botiguer s'hagués de desplaçar per demanar-ho.

- Aplicacions per períodes especials:

En determinades èpoques del any, s'altera el funcionament normal de la pastisseria i la producció es centra en un tipus de producció. Seria útil realitzar una aplicació específica per controlar la gran demanda d'aquests productes especials i el seu emmagatzematge.

- Millorar la pagina Web:

Estaria bé afegir més utilitats a la pàgina web perquè el propietari disposi d'informació actualitzada dels esdeveniments que es produeixin. D'aquesta manera, hi podria accedir des de qualsevol ordinador.

6.2. Conclusions

Aquest projecte ha estat de gran utilitat per enriquir el meu coneixement sobre el procés de desenvolupament d'un programari específic per una empresa. Com que he estudiat ET Informàtica de Sistemes, és un tema que no s'ensenya i sempre és interessant pel meu futur professional. Els coneixements que he adquirit sobre l'anàlisi, el disseny i el desenvolupament son fruit d'una optativa que he escollit: Anàlisis i Disseny Orientat a Objectes (ADOO).

Realitzar el projecte ha estat molt interessant perquè he pogut accedir a les necessitats reals de l'empresa he hagut d'adaptar el programa als requeriments sol·licitats. També he obtingut coneixements sobre com es gestiona una pastisseria utilitzant recursos informàtics bàsics que qualsevol persona pot tenir al seu ordenador personal. A més, he pogut veure alguna utilitat desenvolupada per ells mateixos amb llenguatges a nivell de consola.

Aquest programa estarà en període de proves pel propietari per valorar si en un futur el pot utilitzar, si es decideix a comprar el nou equipament. També haurà de tenir en compte altres utilitats que els interessaria tenir.

La principal dificultat ha estat la limitada formació sobre l'anàlisi, disseny i desenvolupament. Tampoc no havia efectuat mai una comunicació entre un programa i una base de dades. Altres problemes han sorgit per incompatibilitat entre el compilador i altres utilitats (com els *Reports*). Per acabar amb les dificultats, cal destacar la manca de temps real per realitzar totes les utilitats necessàries pel client ja que, tot i haver començat l'anàlisi a partir del febrer, per problemes d'horaris deguts a l'estudi de la carrera no m'hi he pogut centrar fins haver acabat els exàmens de juny.

Per acabar la memòria, cal destacar que aquest projecte ha estat molt útil per poder complementar la meva formació sobre temes que no es tracten a classe i són necessaris per l'experiència professional. També remarcar que ha estat un cas real i, per tant, les dificultats sorgides són molt diferents a les que ens podem trobar en qualsevol altre pràctica realitzada durant la carrera.

7. Bibliografia

CHARTE OJEDA, Francisco. *PHP 5*. Madrid: Anaya, cop. 2005. ISBN 84-415-1796-7

GÓMEZ VILLADANGOS, Montserrat. *Fitxers i bases de dades*. [Vic]: Universitat de Vic. Escola Politècnica Superior, [s. a.].

Proteger carpetas con .htaccess y .htpasswd [En línia]. [s. l.]: Cristalab, 2007. <http://www.cristalab.com/tutoriales/proteger-carpetas-con-.htaccess-y-.htpasswd-c213l/> [Consulta: 3 setembre 2009]

RUMBAUGH, Jacobson. *El lenguaje unificado modelado. Manual de referencia*. [s. l.]: Addison – Wesley, [s. a.].