Integrantes : Álvaro Elizalde, Daniel Barroso, Diego Flores, Gonzalo Casado, Isabel Zamarrón, José Antonio Tortosa, Laura Díez, Mario Campos, Melany Chicaiza, Nerik Forcada, Santiago Caudillo y Santiago González

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito

Nuestra SRS tiene el propósito de desarrollar una aplicación orientada a la gestión de un restaurante, incluyendo todos los requisitos que son necesarios para poder elaborar el proyecto, así como la funcionalidad, las prestaciones, restricciones del diseño, atributos o interfaces externas. El principal objetivo es proporcionar un mayor conocimiento del proyecto, para la asignatura de MS.

Nuestra principal audiencia serán los clientes que soliciten el servicio del restaurante y nosotros nos encargamos de coordinar la logística para que los servicios se presten adecuadamente.

1.2 Alcance

La aplicación desarrollada tiene el objetivo de crear, modificar, representar y eliminar la información de las mesas,producto,plato,empleado,turno y ventas que constituyen nuestro restaurante.

1.3 Definiciones, acrónimos y abreviaturas

- SRS (Software Requirements Specification): documentación de los requisitos esenciales (funciones, restricciones del diseño, atributos y funcionamiento) del proyecto y las interfaces externas.

- SQL (Structured Query Language): Lenguaje de consulta estructurada.

- UML: Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

- Módulo: Parte de un programa lógicamente separable, por ejemplo por las entidades que gestionan la aplicación.

- Plato: Alimento preparado o cocinado para ser servido a los clientes,en nuestro caso como nos centramos en un buffet sería una serie indefinida de plato que el cliente pida.

-Empleados: Personas que se encargan de cocinar cada plato como de servirlos en la mesa,cada uno tiene un turno que debe cumplir.

- Mesas: El lugar en donde a los clientes les sirven sus platos los camareros y una vez acabado el pedido se podrá asignar dicha mesa a otros clientes.

- Turno: Horario que se le asigna a cada empleado además de quedar reflejado en su nómina.

- Producto: Cosa producida natural o artificialmente, o resultado de un trabajo u operación. Es utilizado para la composición de cada plato sin ellos no se puede realizar.

- Venta: es la acción de transferir la propiedad de un bien o servicio a cambio de un pago. Esta transacción se realiza entre un vendedor y un comprador, y puede ser realizada en un establecimiento físico o en línea. La venta es una parte importante del comercio y de la economía, ya que permite a las empresas obtener ingresos y a los consumidores adquirir bienes y servicios.

1.4 Referencias

- IEEE Std 830-1998
- IEEE Std 610.12-1990
- Transparencias de la asignatura

1.5 Resumen

Este documento podría resumirse en 3 puntos importantes:

1. El primero sería la introducción, en la cual describimos a rasgos generales como queremos enfocar el tema elegido sin detallarnos en profundidad en cada punto tratado ya que eso lo haremos más adelante.

2. El segundo sería la descripción general, en donde mencionaremos las interfaces que vamos a utilizar, las restricciones necesarias para el buen funcionamiento de nuestra app y una lista de las funciones que queremos realizar para cada módulo que hemos elegido previamente.

3. El tercer punto desarrolla todos los temas que anteriormente solo hemos hecho referencia pero no hemos detallado, como puede ser el caso de las funciones de cada módulo, que ahora sí serán descritas una a una y también se explicará la relación que tienen entre sí cada módulo, además de ampliar la información de las restricciones, las interfaces y los requisitos.

2. DESCRIPCIÓN GENERAL

2.1 Perspectiva de producto

El principal y único objetivo del producto es facilitar la gestión de un restaurante. Nuestro producto no forma parte de un sistema mayor ni es ninguna ampliación de otro producto existente, es decir, es un producto independiente

- **Interfaces del sistema:** Será necesario una base de datos para el almacenamiento de la información y para su posterior utilización por parte de los gestores.

- **Interfaces de usuario:** Consistirá en un menú donde se podrá acceder a todos los módulos anteriormente mencionados de forma fácil e intuitiva, es decir, gráfico.

- **Interfaces software:** Hacemos uso de JDBC (Java Database Connectivity).

2.2 Funciones del producto

2.1.1. Turnos

1.1) Alta de turno

1.2) Baja de turno

1.3) Modificación de turno

1.4) Consulta turno

1.5) Listar turnos

1.6) Nómina de turno

2.1.2. Mesas

2.1) Alta mesa

2.2) Baja mesa

2.3) Modificación mesa

2.4) Consulta de mesa

2.5) Listar mesas

2.6) Vincular mesa con venta

2.7) Desvincular mesa con venta

2.8) Mostrar mesas por empleado

2.1.3. Platos

3.1) Alta plato

3.2) Baja plato

3.3) Modificación plato

3.4) Consulta plato

3.5) Listar platos por productos

3.6) Vincular plato a producto

3.7) Desvincular plato a producto

2.1.4. Productos

4.1) Alta productos

4.2) Baja productos

4.3) Consulta productos

4.4) Listar productos por alergias

4.5) Modificar productos

4.6) Mostrar productos por plato

2.1.5. Empleados

5.1) Alta empleados

5.2) Modificación empleados

5.3) Baja empleados

5.4) Consulta empleados

5.5) Listar empleados

5.6) Mostrar mesas por empleados

5.7) Mostrar empleados por turno

5.8) Nómina de empleados

2.1.6. Ventas

6.1) Bajas venta

6.2) Consulta venta

6.3) Modificación venta

6.4) Listar ventas

6.5) Devolución venta

6.6) Abrir venta

6.7) Cerrar venta

6.8) Añadir plato

6.9) Eliminar plato

6.10) Mostrar ventas por cliente

6.11) Mostrar ventas por mesa

2.2. Características de usuario

Es una aplicación para un único usuario (Gestor del Restaurante)

- **Gestor:**

  Nivel educativo: Estudios secundarios o superiores.

  Experiencia: Al menos 1 año de experiencia como gestor en otras instituciones similares a la nuestra.

  Capacidades: Conocimiento del sistema para el asesoramiento de los clientes, así como un mínimo conocimiento del manejo de Windows.

2.3. Restricciones

Políticas de regulación:
Seguiremos los Requisitos Software (ERS) según la última versión del estándar IEEE 830 de 1998.

Limitaciones hardware:
Nuestros servidores tienen una capacidad limitada que permite un máximo de 50 usuarios registrados.

Interfaces con otras aplicaciones:
Nuestra aplicación no depende de ninguna otra, por lo que no existen restricciones.

Operaciones en paralelo:
Hay paralelismo ya que varios usuarios pueden estar solicitando una reserva o sus platos simultáneamente.

Funciones de auditoría:
Se encargan de vigilar el cumplimiento de los controles internos diseñados por la gerencia, y

agrega valor a la organización dando recomendaciones para corregir las debilidades de control interno y mejorar la eficacia de los procesos.

Funciones de control:
Permiten señalar el desvío entre lo planeado y lo realizado para, de esta forma, corregir nuestras acciones con el fin de lograr los objetivos fijados.

Requisitos de lenguaje de alto nivel:
Los lenguajes de alto nivel que utilizaremos serán Java y SQL.

Protocolos de signal handshake:
Tienen lugar cuando un equipo está a punto de comunicarse con un dispositivo exterior para establecer las normas de comunicación. Cuando un ordenador se comunica con otro dispositivo como un módem o una impresora se necesita realizar un establecimiento con este para crear una conexión.

Requisitos de fiabilidad:
Nuestra aplicación requiere conexión a internet, aunque sigue siendo segura y fiable.

Criticidad de la aplicación:
Permite establecer las prioridades de procesos, sistemas y equipos, creando una estructura que facilita la toma de decisiones acertadas y efectivas, direccionando el esfuerzo y los recursos en áreas donde sea más importante mejorar la fiabilidad.

Consideraciones de robustez y seguridad:
Se encarga de comprobar si la aplicación se comporta de forma razonable aún en circunstancias que no fueron anticipadas en la especificación de requerimientos. Por ejemplo cuando encuentra datos de entrada incorrectos o algún mal funcionamiento del hardware.

3. REQUISITOS ESPECÍFICOS

3.1 Funciones

DATOS VENTA: (id, fecha, cliente, precio, cantidad)

DATOS EMPLEADO: (id, nómina, horas, turno)

DATOS MESAS: (número, capacidad,id mesa)

DATOS PLATO: (nombre, precio, alergias, stock,id plato)

DATOS PRODUCTO: (nombre, características,id producto)

DATOS TURNO: (fecha de entrada, fecha de salida, id turno)

1.1.1. Módulo Turnos

1.1.1.1. ALTA TURNOS

Descrito por Gonzalo Casado Valcárcel
Prioridad: Alta

Estabilidad: Alta
Descripción: Registra un nuevo turno, es decir las horas en las que trabaja cada empleado
Actor: Gestores
Entrada: Hora de entrada, hora de salida y nombre
Salida: ID del plato
Origen: Teclado
Destino: Base de datos
Acciones: Si los datos no son válidos se informará de ello, en caso de que sean válidos se comprobará si existe ya en la base de datos, si no existe se dará de alta y se informará de la correcta alta de turno, en caso contrario se informará de que el turno ya existe en la base de datos.
Precondición: No puede haber varios turnos con el mismo ID
Postcondición: Que se hayan escrito bien los datos de entrada, si es así, se crea en la base de datos con el ID que le toque

1.1.1.2. BAJA TURNOS

Descrito por Gonzalo Casado Valcárcel
Prioridad: Media
Estabilidad: Alta
Descripción: Elimina un turno de la base de datos, si lo consigue, significa que ahora este turno pasa a estar inactivo y si no lanza un mensaje de éxito y si no, de error
Actor: Gestores
Entrada: ID producto
Salida: éxito/no éxito
Origen: Teclado
Destino: Base de datos
Acciones: Se busca el turno con ese ID, si no lo encuentra manda mensaje de error. Si lo encuentra, mira el booleano Activo, si no está activo, manda un mensaje de que ya estaba inactivo y si está activo cambia a false y se manda mensaje de éxito
Precondición: No puede haber varios turnos con el mismo ID
Postcondición: Que no haya turnos repetidos en la base de datos, si es así, se modifica en la base de datos

1.1.1.3. MODIFICACIÓN TURNOS

Descrito por Gonzalo Casado Valcárcel
Prioridad: Media
Estabilidad: Alta
Descripción: Se modifica el horario de un turno.

Actor: Gestores
Entrada: Las nuevas fechas de entrada y de salida y el ID del turno a modificar
Salida: Turno modificado
Origen: Teclado
Destino: Base de datos
Acciones: Se comprueba si los datos son válidos, si no, lanza mensaje de error, si sí, busca el turno con ese ID y lo modifica, si no lo encuentra lanza mensaje de error
Precondición: No puede haber varios turnos con el mismo ID
Postcondición: Que no haya turnos repetidos en la base de datos, si es así, se modifica en la base de datos

1.1.1.4. CONSULTAR TURNOS

Descrito por Gonzalo Casado Valcárcel
Prioridad: Alta
Estabilidad: Alta
Descripción: Accede a los datos del empleado que cumpla con los requisitos descritos por los gestores
Actor: Gestores
Entrada: Nombre del empleado
Salida: Datos del empleado
Origen: Teclado
Destino: Base de datos
Acciones: Se mira el nombre del empleado y se busca en la base de datos si existen empleados con ese nombre, si existen se listan con todos los datos de cada uno. Si no, se manda un mensaje de error
Precondición: No puede haber varios turnos con el mismo ID
Postcondición: Que no haya turnos repetidos en la base de datos, si es así, se modifica en la base de datos.

1.1.1.5. LISTAR TURNOS

Descrito por Gonzalo Casado Valcárcel
Prioridad: Alta
Estabilidad: Alta
Descripción: Sirve para mostrar todos los turnos que hay registrados en la base de datos y cada una de sus horas de salida y entrada, además de si está activo o no.
Actor: Gestores
Entrada:
Salida: una lista de los turnos de la base de datos
Origen: Teclado

Destino: Base de datos
Acciones: Se recorre la lista de turnos que hay y accede a cada uno de los datos de dentro de cada turno para poder mostrarlos en la interfaz, si no hay ningún turno en la lista se manda un mensaje informando sobre ello
Precondición: No puede haber varios turnos con el mismo ID
Postcondición: Que no haya turnos repetidos en la base de datos, si es así, se modifica en la base de datos.

1.1.1.6. NÓMINA DE TURNO

Descrito por Gonzalo Casado Valcárcel
Prioridad: Alta
Estabilidad: Alta
Descripción: Se calcula el salario del empleado en función de su especialización (cocinero o camarero) y su turno.
Actor: Gestores
Entrada: Id de empleado
Salida: Salario correspondiente al trabajador
Origen: Teclado
Destino: Base de datos
Acciones: Con el Id, busca al empleado en la base de datos, si existe, mira si es cocinero o camarero. Si no, lanza un mensaje de error para indicar que no existe. En cuanto se sabe si es camarero o cocinero, se mira el turno del trabajador y se calcula el salario para el trabajador indicado
Precondición: No puede haber varios empleados con el mismo Id
Postcondición: Que no haya empleados repetidos en la base de datos, si es así, se modifica en la base de datos

1.1.2. Módulo Mesas

1.1.2.1. ALTA MESAS

Descrito por Daniel Barroso Casado
Prioridad: Alta
Estabilidad: Alta
Descripción: Registramos mesas del restaurante en la base de datos, aportando información como su capacidad y el número de mesas correspondiente al conjunto

Actor: Gestores
Entrada: ID de la mesa, nº de mesas y capacidad
Salida: ID de la mesa
Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos de la mesa y comprueba que son sintácticamente correctos, en caso de no serlos envía un mensaje de error, y si son correctos busca si hay alguna otra mesa con el mismo ID en la base de datos. Si se encuentra la mesa en la base de datos, comprobaremos si el booleano activo se encuentra en true o en false. En el caso de que el booleano esté en true informaremos de que la mesa ya se encuentra dada de alta en la base de datos, y en el segundo caso cambiaremos el booleano de la mesa a true. En caso de que la mesa no se encuentre en la base de datos la insertamos.
Precondición: Que no haya mesas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya mesas repetidas en la base de datos, si éxito, añadir esa mesa a la base de datos.

1.1.2.2. BAJA MESAS

Descrito por Daniel Barroso Casado
Prioridad: Alta
Estabilidad: Alta
Descripción: Damos de baja una mesa de la base de datos
Actor: Gestores
Entrada: ID de la mesa
Salida: ID de la mesa
Origen: Teclado
Destino: Base de datos
Acciones: Comprobamos que la mesa existe en la base de datos. En caso de que no exista, enviamos un mensaje de error diciendo que la mesa no existe. En caso de que sí existe, si el booleano activo se encuentra en false, enviamos un mensaje de error indicando que la mesa ya se encuentra dada de baja, y si el booleano activo se encuentra en true, damos de baja la mesa cambiando el booleano a false y se informa de la correcta baja del plato
Precondición: Que no haya mesas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya mesas repetidas en la base de datos, si éxito, eliminamos la mesa

1.1.2.3. MODIFICACIÓN MESAS

Descrito por Daniel Barroso Casado

Prioridad: Alta
Estabilidad: Alta
Descripción: Modificamos la mesa de la base de datos, porque queremos modificar el número de mesas que la componen o su capacidad
Actor: Gestores
Entrada: ID de la mesa, nº de mesas y capacidad
Salida: mesa modificada
Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos de la mesa y comprueba si existe en la base de datos. En caso de que no exista, se lanza un mensaje de error diciendo que la mesa no se puede modificar porque no existe. En caso de que exista, se modificará la mesa y se informará de su correcta modificación
Precondición: Que no haya mesas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya mesas repetidas en la base de datos, si éxito, modificamos la mesa

1.1.2.4. CONSULTAR MESAS

Descrito por Daniel Barroso Casado
Prioridad: Alta
Estabilidad: Alta
Descripción: Consultamos los datos de una mesa en concreto, aquel del que introducimos su ID
Actor: Gestores
Entrada: ID de la mesa
Salida: datos de la mesa
Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos de la mesa y comprueba si existe en la base de datos. En caso de que no exista, se lanza un mensaje de error diciendo que la mesa no se puede mostrar porque no existe. En caso contrario, se muestra toda la información de la mesa solicitada
Precondición: Que no haya mesas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya mesas repetidas en la base de datos, si éxito, mostramos la información de la mesa

1.1.2.5. LISTAR MESAS

Descrito por Daniel Barroso Casado
Prioridad: Alta

Estabilidad: Alta
Descripción: Accedemos a todas las mesas que tenemos en la base de datos
Actor: Gestores
Entrada:
Salida: lista de todas las mesas
Origen: Teclado
Destino: Base de datos
Acciones: Comprobamos que existe alguna mesa en la base de datos, si no existe mandamos una mensaje informando de ello, y en caso de que exista mostramos el listado de las mesas que exitan
Precondición: Que no haya mesas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya mesas repetidas en la base de datos, si éxito, mostramos las mesas

1.1.2.6. VINCULAR MESA CON VENTA

Descrito por Daniel Barroso Casado
Prioridad: Alta
Estabilidad: Alta
Descripción: Asocia la mesa con una venta
Actor: Gestores
Entrada: ID de la mesa e ID de la venta
Salida: ID de la mesa e ID de la venta
Origen: Teclado
Destino: Base de datos
Acciones: Recibimos el ID de la mesa y comprobamos que existe en la base de datos. Si la mesa no existe en la base de datos,mandamos un mensaje de error informando de ello y en caso contrario, comprobamos que existe una venta con el ID proporcionado. En caso de que no exista, mandamos un mensaje de error informando de ello y en caso contrario, comprobamos que esa mesa y esa venta no están ya vinculados. Si ya están vinculados, enviamos un mensaje de error informando de ello y en caso contrario, vinculamos la mesa con la venta y mandamos un mensaje de éxito de la operación
Precondición: Que no haya mesas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya mesas repetidas en la base de datos, si éxito, vinculamos la mesa y la venta

1.1.2.7. DESVINCULAR MESA CON VENTA

Descrito por Daniel Barroso Casado
Prioridad: Alta

Estabilidad: Alta
Descripción: Asocia la mesa con una venta
Actor: Gestores
Entrada: ID de la mesa e ID de la venta
Salida: ID de la mesa e ID de la venta
Origen: Teclado
Destino: Base de datos
Acciones: Recibimos el ID de la mesa y comprobamos que existe en la base de datos. Si la mesa no existe en la base de datos,mandamos un mensaje de error informando de ello y en caso contrario, comprobamos que existe un empleado con el ID proporcionado. En caso de que no exista, mandamos un mensaje de error informando de ello y en caso contrario, comprobamos que esa mesa y esa venta no están ya vinculados. Si no están vinculados, enviamos un mensaje de error informando de ello y en caso contrario, desvinculamos la mesa con la venta y mandamos un mensaje de éxito de la operación
Precondición: Que no haya mesas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya mesas repetidas en la base de datos, si éxito, desvinculamos la mesa y la venta

1.1.2.8. MOSTRAR MESAS POR EMPLEADO

1.1.3. Módulo Platos

1.1.3.1. ALTA PLATOS

Descrito por Laura Díez Matarranz
Prioridad: Alta
Estabilidad: Alta
Descripción: Registramos los platos que pueden escoger los clientes en la carta,pero no podriamos servirlo sino tenemos todos los productos que se necesitan para hacer el plato
Actor: Gestores
Entrada: Datos del plato
Salida: La información del plato que se ha buscado.
Origen: Teclado
Destino: Base de datos

Acciones: Recibe los datos del plato y comprueba si son correctos,de no serlos se informaría de que no son válidos,en cambio si son correctos buscamos si el plato ya está en la base de datos
Si se encuentra en la base de datos debería comprobarse si el booleano está activo o no,si esta true se informa de que está en la base de datos y si está a false se pone a true para indicar que se da de alta y se informa de su correcta dada alta
Pero por el contrario si no se encuentra en la base de datos se comprobaría que todos los productos que componen el plato están en la base de datos
Si alguno no está se darán de alta y se registraría el plato además de poner el booleano a true para indicar que está activo y se informaría de su correcta dada de alta
Precondición: Que no haya platos repetidos en la base de datos, es decir, que el ID no esté repetido ni que esté el nombre repetido
Postcondición: Que no haya platos repetidos en la base de datos, si éxito, añadir dicho plato en la base de datos

1.1.3.2. BAJA PLATOS

Descrito por Laura Díez Matarranz
Prioridad: Alta
Estabilidad: Alta
Descripción: Da de baja a un plato de la base de datos
Actor: Gestores
Entrada: Datos platos
Salida: ID del plato
Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos del plato y comprueba si son correctos,de no serlos se informaría de que no son válidos,en cambio si son correctos buscamos si el plato ya está en la base de datos
Si se encuentra en la base de datos debería comprobarse si el booleano está activo o no,si esta true se cambia a false y se informa de la correcta baja y si está a false se informa de que ese plato ya estaba dado de baja
Pero por el contrario si no se encuentra en la base de datos se informaría de que no se puede dar de baja un plato que no está en la base de datos
Precondición: Que no haya platos repetidos en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya platos repetidos en la base de datos, si éxito, eliminamos al plato

1.1.3.3. MODIFICACIÓN PLATOS

Descrito por Laura Díez Matarranz

Prioridad: Alta
Estabilidad: Alta
Descripción: Modificamos el plato en la base de datos, es decir,puede haber un cambio en los productos de su elaboración,puede darse el caso o bien porque nos hemos quedado sin el producto pero no queremos quitarlo de la carta o por querer un cambio en el plato
Actor: Gestores
Entrada: El atributo a modificar
Salida: plato modificado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos que queremos modificar y se comprueba si son válidos,si no lo son se informa de ello.En el caso de que sean válidos se comprueba que el plato esté en la base de datos ,si no esta no puede realizarse la modificación pero, si se encuentra en la base de datos se modificará y se avisará de su correcta modificación
Precondición: Que no haya platos repetidos en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya platos repetidos en la base de datos, si éxito, modificar dicho plato en la base de datos

1.1.3.4. CONSULTAR PLATOS

Descrito por Laura Díez Matarranz
Prioridad: Media
Estabilidad: Alta
Descripción: Accede únicamente a los datos de un plato, aquel que cumpla con lo que buscan los gestores
Actor: Gestores
Entrada: El o los atributos del plato por los que se quiera filtrar para buscar
Salida: ID del plato
Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos del plato y se comprueba si son válidos, de no serlo se informará de ello pero si son correctos se buscará el plato con esas características.
Puede darse el caso de que no se encuentre ningún plato y se informe de que su consulta a sido fallida o puede ocurrir que se que haya platos con esas características y se muestren todos ellos además de informar de su correcta consulta
Precondición: Que no haya platos repetidos en la base de datos, es decir, que el ID no esté repetido

Postcondición: Que no haya platos repetidos en la base de datos, si existo, hacer la consulta de platos

1.1.3.5. LISTAR PLATOS POR PRODUCTO

Descrito por Laura Díez Matarranz
Prioridad: Alta
Estabilidad: Alta
Descripción: Acceder a todos los platos que tenemos en nuestra base de datos y los productos que contiene cada plato
Actor: Gestores
Entrada:
Salida: Todas los platos que tenemos en nuestra base de datos con sus productos correspondientes
Origen: Teclado
Destino: Base de datos
Acciones: Busca en la base de datos todos los platos que tenemos y los muestra e informa de que se ha realizado bien la función o que no tengamos ningún plato y se informe de ello
Precondición: Que no haya platos repetidos en la base de datos, es decir, que el ID no esté repetido.
Postcondición: Que no haya platos repetidos en la base de datos, si éxito, muestran todas las actividades en una lista.

1.1.3.6. VINCULAR PLATO A PRODUCTO

Descrito por Laura Díez Matarranz.
Prioridad: Media
Estabilidad: Alta
Descripción: Asocia el plato con el producto
Actor: Gestores
Entrada: Datos e ID del plato y del producto
Salida: ID del plato y del producto
Origen: Teclado
Destino: Base de datos
Acciones: Recibir el id del plato y comprobar si es valido, de serlo buscar en la base de datos dicho id,si existe entonces elegir el producto con el que lo quiero vincular y pasarle el id de ese producto y realizar sus correspondientes comprobaciones y búsquedas para saber si puede vincularse este producto
Una vez que sabemos que tanto el plato como el producto existen en la base de datos puede darse el caso de que ya estén vinculados entonces se avisará de ello, pero por el contrario puede que no estén vinculados entonces se vincularían y se informaría de su correcta vinculación

Precondición: ID único del plato y del producto en la base de datos
Postcondición: ID único del plato y del producto en la base de datos, si éxito, vincular el plato al producto

1.1.3.7. DESVINCULAR PLATO A PRODUCTO

Descrito por Laura Díez Matarranz
Prioridad: Media
Estabilidad: Alta
Descripción: La retirada del plato a un producto con el que está vinculado
Actor: Gestores
Entrada: ID del plato y el del producto vinculado
Salida: Un entero (int) que me devuelva si está vinculado o no previamente
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el id del plato y del producto se comprueba si son válidos si no lo son se informa de ello,pero si son válidos se busca el id del producto en la base de datos si existe se buscaría el del plato,de existir se comprobaría si están vinculados,de estarlo entonces se desvinculan el plato del producto y se informaría de su correcta desvinculación
Precondición: ID único del plato y del producto en la base de datos.
Postcondición: ID único del plato y del producto en la base de datos, si éxito, desvincular el plato al producto.

1.1.4. Módulo Productos

1.1.4.1. ALTA PRODUCTO

Descrito por Nerik Forcada García.
Prioridad: Alta
Estabilidad: Alta
Descripción: Registra un nuevo producto en la base de datos
Actor: Gestores
Entrada: Datos del producto
Salida: ID del producto
Origen: Teclado
Destino: Base de datos
Acciones: Se comprueba que los datos del producto sean correctos, en caso de que no se informará al usuario de ello, en caso de que sí se comprobará si existe

o no ese producto en la base de datos, en caso afirmativo se informará de que el producto ya existe en la base de datos, en caso contrario se dará de alta el producto y se informará del correcto registro
Precondición: Que no haya productos repetidos en la base de datos, es decir, que el nombre e ID no estén repetidos.
Postcondición: Que no haya productos repetidos en la base de datos,si éxito, añadir dicho producto a la base de datos

1.1.4.2. BAJA PRODUCTO

Descrito por Nerik Forcada García
Prioridad: Media
Estabilidad: Alta
Descripción: Elimina un producto de la base de datos a no ser que el producto se encuentre vinculado a algún plato que se encuentre activo
Actor: Gestores
Entrada: Datos del producto
Salida: ID del producto
Origen: Teclado
Destino: Base de datos
Acciones: Se comprueba que los datos del producto sean válidos, en caso de que no lo sean se informa de que ello, en caso contrario se revisa que el producto exista en la base de datos y si existe y estaba activo se da de baja cambiando el activo a false, en caso de que no exista se informará de ello y lo mismo en caso de que el activo estuviera a false, informando de que el producto ya estaba dado de baja
Precondición: Que no haya productos repetidos en la base de datos, es decir, que el nombre e ID no estén repetidos.
Postcondición: Que no haya productos repetidos en la base de datos y no se encuentre en ningún plato activo, si éxito, eliminamos el producto.

1.1.4.3. CONSULTA PRODUCTO

Descrito por Nerik Forcada García.
Prioridad: Alta
Estabilidad: Alta
Descripción: Accede a los datos del producto que cumpla con los requisitos descritos por los gestores.
Actor: Gestores
Entrada: Nombre del producto
Salida: Datos del producto
Origen: Teclado
Destino: Base de datos

Acciones: Se comprueba que los datos introducidos sean correctos, en caso de que no se informará de que el producto no existe en la base de datos, en caso de que el producto si exista en la base de datos se comprobará si está activo. Si no está activo se informará de que no se puede consultar el producto ya que se dió de baja, en cambio si está activo se mostrarán los datos del producto
Precondición: Que no haya productos repetidos en la base de datos, es decir, que el nombre e ID no estén repetidos.
Postcondición: Que no haya productos repetidos en la base de datos, si éxito, hacer la consulta del producto.

1.1.4.4. LISTAR PRODUCTOS

Descrito por Nerik Forcada García
Prioridad: Media
Estabilidad: Alta
Descripción: Acceder a todos los productos que tenemos en nuestra base de datos
Actor: Gestores
Entrada:
Salida: Todas los productos que tenemos en nuestra base de datos
Origen: Teclado
Destino: Base de datos
Acciones: Se comprueba que los datos de la base de datos son correctos y si existen productos, en caso contrario se informará de que los datos no son válidos o no existen productos en la base de datos, sino se mostrarán todos los productos
Precondición: Que no haya productos repetidos en la base de datos, es decir, que el nombre e ID de los productos no estén repetidos.
Postcondición: Que no haya productos repetidos en la base de datos, si éxito, muestran todos los productos en una lista.

1.1.4.5. MODIFICAR PRODUCTOS

Descrito por Nerik Forcada García
Prioridad: Alta
Estabilidad: Alta
Descripción: Modifica el producto en la base de datos, es decir, puede haber un cambio en las características del producto
Actor: Gestores
Entrada: Datos a modificar del producto
Salida: Producto del usuario modificado
Origen: Teclado
Destino: Base de datos

Acciones: Se comprueba que el producto sea válido y exista en la base de datos, en caso de que no lo sea se informará de ello, en caso contrario se modificarán los atributos deseados del producto.
Precondición: Que no haya productos repetidos en la base de datos, es decir, que el nombre e ID no estén repetidos.
Postcondición: Que no haya productos repetidos en la base de datos, si éxito, modificar dicho producto en la base de datos.

1.1.4.6. MOSTRAR PRODUCTOS POR PLATO

Descrito por Nerik Forcada García.
Prioridad: Alta
Estabilidad: Alta
Descripción: Muestra los productos que contiene el plato escogido
Actor: Gestores
Entrada: Datos del plato
Salida: Nombre de los productos del plato
Origen: Teclado
Destino: Base de datos
Acciones: Se comprueba que los datos del plato sean válidos, en caso de que no lo sean se informará de ello, en caso contrario se comprobará si el plato existe en la base de datos, si no existe se informará de que no existe, en caso de que sí exista se comprobará que esté activo, si no lo está se informará de que el plato no se encuentra activo, si lo está finalmente se mostrarán los productos activos que componen el plato.
Precondición: Que no haya platos ni productos repetidos en la base de datos, es decir, que el ID del plato, ID y nombre del productos no estén repetidos.
Postcondición: Que no haya platos ni productos repetidos en la base de datos, si éxito, mostrar los productos que dicho plato tiene en la base de datos.

1.1.5. Módulo Empleados

1.1.5.1. ALTA EMPLEADOS

Descrito por Melany Daniela Chicaiza Quezada
Prioridad: Alta
Estabilidad: Alta
Descripción: Registra un nuevo empleado en la base de datos
Actor: Gestores

Entrada: Datos del empleado
Salida: ID del empleado
Origen: Teclado
Destino: Base de datos
Acciones: recibe los datos del empleado y comprueba que sean correctos. De no serlo se informaría de que no son válidos, si lo son, se revisa que no exista ya en la base de datos. Si se encuentra en la base de datos se comprueba que el booleano de activo está a true. Si está a false se pone a true para indicar su alta y se informa de ello, si está a true se informa de que sí se encontraba en la base de datos
Precondición: Que no haya empleados repetidos en la base de datos, es decir, que ni el nombre ni el id estén repetidos
Postcondición: Que no haya empleados repetidos en la base de datos, si éxito, añadir dicho empleado a la base de datos

1.1.5.2. BAJA EMPLEADOS

Descrito por Melany Daniela Chicaiza Quezada
Prioridad: Alta
Estabilidad: Alta
Descripción: Elimina un empleado de la base de datos, si no tiene mesas vinculadas
Actor: Gestores
Entrada: Datos del empleado
Salida: ID del empleado
Origen: Teclado
Destino: Base de datos
Acciones: recibe los datos del empleado y comprueba que sean válidos (si no lo son se informa de ello). Si son correctos se revisa que el empleado exista en la base de datos. Una vez encontrado en la base de datos se revisa el booleano de activo. Si está a true se cambia a false y se informa de la correcta baja, si está a false se informa de que ya estaba dado de baja
Precondición: Que no haya empleados repetidos en la base de datos, es decir, que ni el nombre ni el id estén repetidos
Postcondición: Que no haya empleados repetidos en la base de datos, si éxito, eliminamos el empleados

1.1.5.3. MODIFICACIÓN EMPLEADOS

Descrito por Melany Daniela Chicaiza Quezada
Prioridad: Alta
Estabilidad: Alta
Descripción: Modificamos el empleado en la base de datos, es decir, puede haber un cambio en sus datos

Actor: Gestores
Entrada: El atributo a modificar
Salida: Empleado modificado
Origen: Teclado
Destino: Base de datos
Acciones: recibe los datos del empleado y revisa que sean válidos (si no lo son se informa de ello). Si son válidos y existe el empleado en la base de datos (si no está en la base de datos se informa de que no puede realizarse la modificación) se modifican los datos y se notifica su modificación correcta
Precondición: Que no haya empleados repetidos en la base de datos, es decir, que ni el nombre ni el ID no esté repetido
Postcondición: Que no haya empleados repetidos en la base de datos, si éxito, modificar dicho empleado en la base de datos

1.1.5.4. CONSULTA EMPLEADOS

Descrito por Melany Daniela Chicaiza Quezada
Prioridad: Media
Estabilidad: Alta
Descripción: Accede a los datos del empleado que cumpla con los requisitos descritos por los gestores
Actor: Gestores
Entrada: El o los atributos del empleado por los que se quiera filtrar para buscar
Salida: ID del empleado
Origen: Teclado
Destino: Base de datos
Acciones: recibe los datos del empleado a consultar y comprueba que sean válidos (de no serlo se informa de ello). Si lo son se comprueba que existe el empleado en la base de datos. Si existe en la base de datos y está activo se muestran todos los datos asociados a dicho empleado y se informa de su correcta consulta (si no se encuentra en la base de datos también se informa de ello)
Precondición: Que no haya empleados repetidos en la base de datos, es decir, que ni el nombre ni el ID esté repetido
Postcondición: Que no haya empleados repetidos en la base de datos, si éxito, hacer la consulta del empleado

1.1.5.5. LISTAR EMPLEADOS

Descrito por Melany Daniela Chicaiza Quezada
Prioridad: Media
Estabilidad: Alta

Descripción: Acceder a todos los empleados que tenemos en nuestra base de datos
Actor: Gestores
Entrada:
Salida: Todas los empleados que tenemos en nuestra base de datos
Origen: Teclado
Destino: Base de datos
Acciones: busca en la base de datos todos los empleados que tenemos, los muestra si están activos y notifica de que ha tenido éxito la operación. En caso de no existir empleados muestra una tabla vacía
Precondición: Que no haya empleados repetidos en la base de datos, es decir, que ni el nombre ni el ID de los empleados estén repetidos
Postcondición: Que no haya empleados repetidos en la base de datos, si éxito, muestran todos los empleados en una lista

1.1.5.6. MOSTRAR EMPLEADOS POR MESA

Descrito por Melany Daniela Chicaiza Quezada
Prioridad: Alta
Estabilidad: Alta
Descripción: Accede a los datos de los empleados que deseen buscar y las mesas vinculadas a él, con el fin de mostrarlo por pantalla
Actor: Gestores
Entrada: El o los atributos de los empleados por los que se quiera filtrar para buscar
Salida: ID de los empleados y las mesas
Origen: Teclado
Destino: Base de datos
Acciones: busca en la base de datos si los empleados existen, si existen y están activos (si no lo están se informa de ello) se mostrarán dichos empleados con mesas correspondientes vinculadas a ellos
Precondición: Que no haya empleados repetidos en la base de datos, es decir, que ni el nombre ni el ID estén repetidos
Postcondición: Que no haya empleados repetidos en la base de datos, si éxito, mostrar toda la lista de empleados y las mesas relacionadas a él

1.1.5.7. MOSTRAR EMPLEADOS POR TURNO

Descrito por Melany Daniela Chicaiza Quezada
Prioridad: Alta
Estabilidad: Alta
Descripción: Accede a los datos del turno que desee buscar y los empleados vinculados a dicho turno, con el fin de mostrarlos por pantalla

Actor: Gestores
Entrada: El o los atributos del turno por el que se quiera filtrar para buscar
Salida: ID de los empleados y los turnos
Origen: Teclado
Destino: Base de datos
Acciones: busca en la base de datos el turno existe, si existe y está activos (si no lo está se informa de ello) se mostrará dicho turno con los empleados correspondientes vinculados a él
Precondición: Que no haya turnos repetidos en la base de datos, es decir, que ni el nombre ni el ID estén repetidos
Postcondición: Que no haya turnos repetidos en la base de datos, si éxito, mostrar toda la lista de empleados y los turnos relacionados a ellos

1.1.5.8. NÓMINA DE EMPLEADOS

Descrito por Melany Daniela Chicaiza Quezada

La operación polimórfica va a ser el cálculo de la nómina de los empleados. Será calculado de forma distinta dependiendo del tipo de empleado, es decir, dependiendo de si es un cocinero o un camarero
Cocinero consta del atributo “especialidad” y camarero del atributo “propinas”
El cálculo de sus nóminas dependerá de lo que cobre por hacer platos de una determinada especialidad si es un cocinero o de las propinas que reciba si es un camarero

1.1.6. Módulo Ventas

1.1.6.1. BAJA VENTA

Descrito por Isabel Zamarrón Mateo
Prioridad: Alta
Estabilidad: Alta
Descripción: Da de baja una venta de la base de datos
Actor: Gestores
Entrada: ID de la venta
Salida: Mensaje de acción realizada con éxito junto al ID de venta correspondiente
Origen: Teclado
Destino: Base de datos

Acciones: Recibe el ID de venta y comprueba si es correcto y si existe en la base de datos, en caso de sí existir, comprueba si se encuentra activa o inactiva, y en el caso de encontrarse activa se procede a dar de baja. En caso contrario se mostraría un mensaje de error.
Precondición: Que no haya ventas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya ventas repetidas en la base de datos, si éxito, eliminamos la venta

1.1.6.2. CONSULTA VENTA

Descrito por Isabel Zamarrón Mateo
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra la información detallada de la venta
Actor: Gestores
Entrada: ID de venta
Salida: Si la venta existe muestra la información detallada de la venta escogida. Al contrario muestra un mensaje informativo.
Origen: Interfaz de venta
Destino: Base de datos
Acciones: Recibe el ID de venta a través de la interfaz, comprueba si existe en la base de datos, en caso de sí existir se genera un TOA con toda la información (ID de venta, fecha, cliente y los platos con su cantidad y precio correspondiente) accediendo tanto a la base de datos de venta como de plato. Una vez se ha realizado esta acción se procede a mostrar toda la información. En caso de que alguna de las operaciones falle, se muestra un mensaje de error
Precondición: Que no haya ventas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya ventas repetidas en la base de datos, si éxito, hacer la consulta de la venta

1.1.6.3. MODIFICACIÓN VENTA

Descrito por Isabel Zamarrón Mateo
Prioridad: Media
Estabilidad: Alta
Descripción: Modifica los atributos de una venta de la base de datos
Actor: Gestores
Entrada: ID de venta y datos que se desea modificar
Salida: Mensaje informativo que indica si la acción se ha realizado con éxito junto al ID de venta correspondiente
Origen: Interfaz de venta

Destino: Base de datos
Acciones: Recibe el ID de la venta a través de la interfaz, comprueba en la base de datos si existe, en caso de sí existir verifica el estado de la venta, en caso de encontrarse activa se procede a guardar la información que se desea modificar, en caso contrario se muestra un mensaje de error y no se realiza ningún cambio en la base de datos
Precondición: Que no haya ventas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya ventas repetidas en la base de datos, si éxito, modificar dicha venta en la base de datos

1.1.6.4. LISTAR VENTAS

Descrito por Isabel Zamarrón Mateo
Prioridad: Alta
Estabilidad: Alta
Descripción: Muestra una lista con todas las ventas registradas en la base de datos.
Actor: Gestores
Entrada:
Salida: Todas las ventas registradas en la base de datos
Origen: Teclado
Destino: Base de datos
Acciones: Busca en la base de datos todas las ventas que se han realizado, las muestra e informa de que se ha realizado bien la función. En caso de no existir ninguna venta se muestra la tabla vacía.
Precondición: Que no haya ventas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya ventas repetidas en la base de datos, si éxito, mostrar todas las ventas registradas

1.1.6.5. DEVOLUCIÓN VENTA

Descrito por Isabel Zamarrón Mateo
Prioridad: Alta
Estabilidad: Media
Descripción: Devuelve una cantidad de platos asociados a una venta.
Actor: Gestores
Entrada: ID de venta, ID de plato, cantidad
Salida: Mensaje de acción realizada con éxito junto al ID de la venta correspondiente
Origen: Teclado

Destino: Base de datos
Acciones: Se introducen el ID de venta, el ID de plato y las cantidad. Comprueba que los datos son correctos y a continuación comprueba si existe una venta y un plato registrados en la base de datos con los ID introducidos.
Comprueba que el plato está incluido en la venta y que su cantidad es mayor que 0. Al contrario se mostrará un mensaje de error.
Si todo funciona correctamente se reducirá el stock del plato con ID plato correspondiente a la venta.
Precondición: Que no haya ventas repetidas en la base de datos, es decir, que el ID no esté repetido
Postcondición: Que no haya ventas repetidas en la base de datos, si éxito, devolver venta

1.1.6.6. ABRIR VENTA

Descrito por Isabel Zamarrón Mateo
Prioridad: Alta
Estabilidad: Alta
Descripción: Crea una nueva venta asociada a una mesa activa
Actor: Gestores
Entrada: ID de mesa
Salida: ID de venta y mensaje de acción realizada con éxito
Origen: Interfaz de venta
Destino: Base de datos
Acciones: Carga la interfaz, recibe el ID de mesa y controla que está existe en la base de datos y que se encuentre activa. Si todo esto es correcto se crea una nueva entrada con los datos introducidos junto con el identificador de esta nueva venta. Si alguna de las comprobaciones falla se muestra un mensaje de error.
Precondición: Que no haya ID de venta e ID de mesa repetidos en la base de datos
Postcondición: Que no haya ID de venta repetidos en la base de datos, si éxito, nueva venta insertada en la base de datos

1.1.6.7. CERRAR VENTA

Descrito por Isabel Zamarrón Mateo
Prioridad: Alta
Estabilidad: Alta
Descripción:
Actor: Gestores
Entrada:

Salida: Si venta tiene al menos un plato se muestra un mensaje éxito en caso contrario se muestra un mensaje de error
Origen: Teclado
Destino: Base de datos
Acciones: Comprueba que todos los datos son correctos y están activos, si la venta contiene al menos un plato. Se cumple que la cantidad de todos los platos solicitados es menor o igual al stock del plato.
En caso afirmativo, se inserta la venta con su precio final calculado. El atributo stock de los platos queda actualizado. Al contrario se muestra un mensaje de error.
Precondición: Que no haya ventas repetidas en la base de datos
Postcondición: Que no haya ventas repetidas en la base de datos, si éxito, cerrar venta

1.1.6.8. AÑADIR PLATO

Descrito por Isabel Zamarrón Mateo
Prioridad: Alta
Estabilidad: Alta
Descripción: Registra los platos que ha escogido un cliente en la venta que se está realizando
Actor: Gestores
Entrada: Datos del plato
Salida: ID de plato, cantidad
Origen: Teclado
Destino: Base de datos
Acciones: Comprueba si el ID de plato es correcto, posteriormente comprueba si se encuentra en la venta actual, en caso de no estar, se añade. Una vez añadido se procede a aumentar la cantidad en uno y a reducir en uno el stock del plato al que corresponde dicho identificador.
Precondición: Que no haya platos repetidos en la base de datos, es decir, que el ID no esté repetido ni que esté el nombre repetido
Postcondición: Que no haya platos repetidos en la base de datos, si éxito, añadir dicho plato en la venta actual

1.1.6.9. ELIMINAR PLATO

Descrito por Isabel Zamarrón Mateo.
Prioridad: Alta
Estabilidad: Alta
Descripción: Quita el plato que ha escogido un cliente de la venta que se está realizando
Actor: Gestores

Entrada: ID de plato, cantidad
Salida:
Origen: Teclado
Destino: Base de datos
Acciones: Comprueba que el ID de plato introducido es correcto, en caso de serlo comprueba si se encontraba añadido a la venta actual. Se procede a reducir la cantidad, si esta pasa a ser menor o igual a 0, se elimina el plato de la venta. Por otra parte, se aumenta el stock del plato en uno. En caso de que el plato no estuviera añadido se muestra un mensaje de error.
Precondición: Que no haya ID de plato repetidos en la base de datos
Postcondición: No se modifica ningún dato en la base de datos

1.1.6.10. MOSTRAR VENTAS POR CLIENTE

Descrito por Isabel Zamarrón Mateo
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra todas las ventas de la base de datos asociadas a un cliente
Actor: Gestores
Entrada: Cliente
Salida: Lista de ventas asociadas a un cliente
Origen: Interfaz de venta
Destino: Base de datos
Acciones: Recibe el cliente a través de la interfaz, se comprueba si este existe en la base de datos, se comprueba si tiene ventas asociadas, en caso de tener ventas se nos mostrará un listado, en caso de no tener ventas se nos mostrará un mensaje de "Cliente sin ventas". En caso de no estar activa o no existir el cliente se nos mostrará un mensaje de error.
Precondición: Que no haya ventas y clientes repetidos en la base de datos
Postcondición: Que no haya ventas repetidas en la base de datos, si éxito, mostrar ventas asociadas al cliente correspondiente.

1.1.6.11. MOSTRAR VENTAS POR MESA

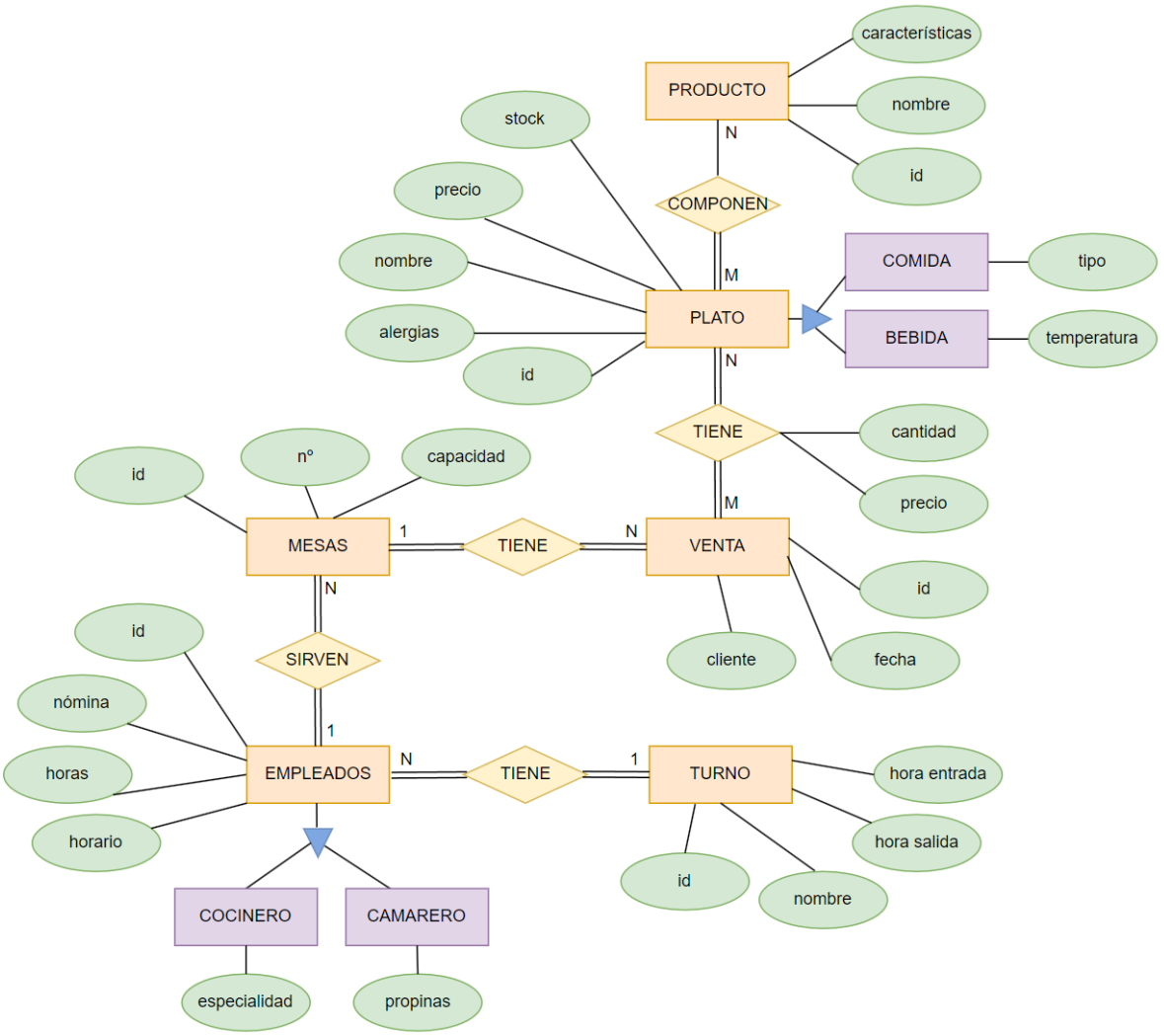
Descrito por Isabel Zamarrón Mateo.
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra todas las ventas de la base de datos asociadas a una mesa.
Actor: Gestores
Entrada: ID de mesa
Salida: Lista de ventas asociadas a una mesa
Origen: Interfaz de venta
Destino: Base de datos

Acciones: Introduce el id de mesa, se comprueba si existe en la base de datos, si existe comprueba que esté activa y en caso afirmativo se comprueba si tiene ventas asociadas, en caso de tener ventas se nos mostrará un listado, en caso de no tener ventas se nos mostrará un mensaje de "Mesa sin ventas". En caso de no estar activa o no existir la mesa se nos mostrará un mensaje de error.
Precondición: Que no haya ventas ni mesas repetidas en la base de datos
Postcondición: Que no haya ventas ni mesas repetidas en la base de datos, si éxito, mostrar toda la lista de ventas y las mesas relacionadas a ella.

3.2 Requisitos de rendimiento

Número de terminales: 1
Número de usuarios: 1
Número de transacciones por segundo: Indefinido
Ordenador y memoria estándar

3.3 Modelo de dominio


características
PRODUCTO
nombre
id
stock
N
precio
COMPONEN
nombre
M
COMIDA
tipo
PLATO
alergias
BEBIDA
temperatura
id
N
TIENE
cantidad
id
n°
capacidad
precio
M
MESAS
1
TIENE
N
VENTA
id
N
id
SIRVEN
cliente
fecha
nómina
1
horas
EMPLEADOS
N
TIENE
1
TURNO
hora entrada
horario
hora salida
id
nombre
COCINERO
CAMARERO
especialidad
propinas


ACADEMIA
flores

1. INTRODUCCIÓN

1.1. Propósito

Nuestra SRS (Software Requirement Specification) tiene el propósito de desarrollar una aplicación orientada a la gestión de una academia educativa, incluyendo todos los requisitos necesarios para llevar a cabo este proyecto. Esto abarcará tanto la funcionalidad esencial como las prestaciones, restricciones de diseño y atributos clave.

Nuestra principal audiencia será la comunidad estudiantil y el personal administrativo de la academia. Nosotros nos encargaremos de la gestión de cursos, inscripciones, evaluaciones y otros aspectos relacionados con la educación.

Esperamos que esta SRS sirva como una guía sólida para el desarrollo del software necesario para mejorar la eficiencia y la experiencia de aprendizaje en la asignatura de MS.

1.2. Alcance

Tenemos como objetivo el manejo de alumnos, sus matrículas, sus cursos y las asignaturas con los profesores que la imparten, además se manejan los clubes de actividades extracurriculares.

1.3. Definiciones, acrónimos y abreviaturas

- SRS (Software Requirements Specification): documentación de los requisitos esenciales (funciones, restricciones del diseño, atributos y funcionamiento) del proyecto y las interfaces externas.

- SQL (Structured Query Language): Lenguaje de consulta estructurada.

- UML: Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

- Módulo: Parte de un programa lógicamente separable, por ejemplo por las entidades que gestionan la aplicación.

- Asignatura: Materia de estudio la cual los alumnos podrán cursar en un año, en un grupo y con un profesor en particular. También cabe destacar que hay dos tipos de asignaturas,

optativas y obligatorias, estas se diferencian por lo que su nombre indica, unas se pueden elegir cursar o no y otras son de carácter obligatorio.

- Matriculable: Es la unión de una asignatura, en un año específico, un grupo determinado y con uno o varios profesores dados. Un alumno se matricula de un matriculable debido a que tiene que elegir todo eso.

- Matrícula: Es la combinación de uno o varios matriculables. Todos los matriculables que elija el alumno componen su matrícula.

- Alumno: Persona que hace y paga la matrícula además de cursar la o las asignaturas y por lo tanto recibir una nota a final de curso.

- Profesor: Persona que se encarga de impartir una asignatura o varias en un año y grupo en particular, también pone la nota. Hay dos tipos de profesores, interino y fijos.

- Año: Especifica en qué año se cursó cada matriculable.

- Grupo: Indica a qué grupo pertenece cada matriculable(A, B C…).

1.4. Referencias

- IEEE Std 830-1998
- IEEE Std 610.12-1990
- Transparencias de la asignatura

1.5. Resumen

Este documento puede resumirse en 3 puntos importantes:

1. El primero es la introducción, donde describimos cómo queremos enfocar el tema elegido sin detallar en profundidad, algo que haremos más adelante.
2. El segundo es la descripción general, en donde mencionamos las interfaces que vamos a usar, las restricciones para el buen funcionamiento del producto y la lista de funciones que tiene cada módulo
3. EL tercer punto desarrolla todos los temas que anteriormente solo hemos hecho referencia pero no hemos detallado, como pueden ser las funciones de cada módulo, las cuales

describiremos, así como las relaciones entre módulos, y ampliaremos la información acerca de restricciones, interfaces y requisitos

2. DESCRIPCIÓN GENERAL

2.1. Perspectiva del producto

El objetivo del producto es la gestión de una academia. El producto no forma parte de un sistema mayor ni es ninguna ampliación de otro producto, ya que es un producto independiente.

- **Interfaces del sistema:** Será necesario una base de datos para el almacenamiento de la información y para su posterior utilización por parte de los gestores.
- **Interfaces de usuario:** Consistirá en un menú donde se podrá acceder a todos los módulos anteriormente mencionados de forma fácil e intuitiva, es decir, gráfico.
- **Interfaces software:** Utilizamos el JDBC.

2.2. Funciones del producto

1) ALUMNOS

1.1) Alta alumno

1.2) Baja alumno

1.3) Modificar alumno

1.4) Consulta alumno

1.5) Listar alumnos

1.6) Query: Listar las notas de un alumno dado

2) PROFESORES

2.1) Alta profesor

2.2) Baja profesor

2.3) Modificación profesor

2.4) Consulta profesor

2.5) Listar profesor

2.6) Listar profesores por matriculable

2.7) Vincular matriculable

3) MATRICULA

3.1) Abrir matrícula

3.2)Vincular matriculable

3.3) Cerrar matrícula

3.4) Baja matrícula

3.5) Modificación matricula

3.6) Consulta matrícula

3.7) Listar matrícula

3.8) Listar matriculables de matrícula

3.9) Consultar Nota por matriculable

3.10) Listar matrículas por alumno

3.11) Calificar Matriculable de Matricula

3.12)Desvincular matriculable

4) ASIGNATURAS

4.1) Alta asignatura

4.2) Baja asignatura

4.3) Modificación asignatura

4.4) Consulta asignatura

4.5) Query: Nota media de una asignatura dada

5) AÑO

5.1) Alta año

5.2) Baja año

5.3) Modificación año

5.4) Consultar año

5.5) Listar año

5.6) Query: Cuántos alumnos hay matriculados dado un año o un rango de años

6) GRUPO

6.1) Alta grupo

6.2) Baja grupo

6.3) Modificación grupo

6.4) Consultar grupo

6.5) Listar grupo

*7) **MATRICULABLE***

7.1) Alta Matriculable

7.2) Baja Matriculable

7.3) Modificación Matriculable

7.4) Consultar Matriculable

7.5) Listar Matriculables

7.6) Listar matriculables por año

7.7) Listar matriculables por grupo

7.8) Listar matriculables por asignatura

2.3. Características de usuario

Es una aplicación para un único usuario (Gestor de la academia)

- **Gestor:**
  - **Nivel educativo:** Estudios secundarios o superiores.
  - **Experiencia:** Al menos 1 año de experiencia como gestor en otras instituciones similares a la nuestra.
  - **Capacidades:** Conocimiento del sistema para el asesoramiento de los clientes, así como un mínimo conocimiento del manejo de Windows.

2.4. Restricciones

Políticas de regulación:

Seguiremos los Requisitos Software (ERS) según la última versión del estándar IEEE 830 de 1998.

Limitaciones hardware:

Nuestros servidores tienen una capacidad limitada que permite un máximo de 50 usuarios registrados.

Interfaces con otras aplicaciones:

Nuestra aplicación no depende de ninguna otra, por lo que no existen restricciones.

Funciones de auditoría:

Se encargan de vigilar el cumplimiento de los controles internos diseñados por la gerencia, y agrega valor a la organización dando recomendaciones para corregir las debilidades de control interno y mejorar la eficacia de los procesos.

Funciones de control:

Permiten señalar el desvío entre lo planeado y lo realizado para, de esta forma, corregir nuestras acciones con el fin de lograr los objetivos fijados.

Requisitos de lenguaje de alto nivel:
Los lenguajes de alto nivel que utilizaremos serán Java y SQL.

Protocolos de signal handshake:
Tienen lugar cuando un equipo está a punto de comunicarse con un dispositivo exterior para establecer las normas de comunicación. Cuando un ordenador se comunica con otro dispositivo como un módem o una impresora se necesita realizar un establecimiento con este para crear una conexión.

Requisitos de fiabilidad:
Nuestra aplicación requiere conexión a internet, aunque sigue siendo segura y fiable.

Criticidad de la aplicación:
Permite establecer las prioridades de procesos, sistemas y equipos, creando una estructura que facilita la toma de decisiones acertadas y efectivas, direccionando el esfuerzo y los recursos en áreas donde sea más importante mejorar la fiabilidad.

Consideraciones de robustez y seguridad:
Se encarga de comprobar si la aplicación se comporta de forma razonable aún en circunstancias que no fueron anticipadas en la especificación de requerimientos. Por ejemplo cuando encuentra datos de entrada incorrectos o algún mal funcionamiento del hardware.

3. REQUISITOS ESPECÍFICOS

3.1. Interfaces externas

DATOS ALUMNOS (DNI, Nombre, Apellido1, Apellido2, edad,teléfono, activo);

DATOS PROFESORES(DNI, Nombre, Apellido1, Apellido2, edad, tipo, *duración, activo);

DATOS PROFESOR-MATRICULABLE(IdProfesor,IdMatriculabe);

DATOS MATRÍCULA(id, precio total, descuento,abierto)

DATOS MATRICULABLE(id, hora, precio, idAsignatura, idGrupo, idAño, activo);

DATOS MATRÍCULA-MATRICULABLE(IdMatricula,IdMatriculable,Nota, activo)

DATOS ASIGNATURA(Id,nombre, tipo, *itinerario, *nivel, activo);

DATOS AÑO(id,Año, activo)

DATOS GRUPOS(id,Letra, activo)

3.2. Funciones

3.2.1. MÓDULO ALUMNOS

3.2.1.1. ALTA ALUMNOS

Descrito por Álvaro Elizalde Romero
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de alta un nuevo alumno de la academia
Actor: Gestor academia
Entrada: Nombre, apellidos, DNI, edad, teléfono
Salida: Mensaje de confirmación
Origen: GUI
Destino: GUI
Acciones: Recibe los datos del alumno y comprueba que tienen un formato correcto. De no serlo se le notifica al cliente. Si son correctos, comprueba si el alumno está dado de alta en la base de datos. De ser así, comprueba si está activo. Si lo estuviese manda un mensaje avisando de que está dado de alta y si no, lo activa. Si el alumno no estuviese inscrito en la base de datos, se le da de alto como activo y se envía un mensaje de confirmación.
Precondición: DNI de alumno no registrado o inactivo en la base de datos
Postcondición: Datos no repetidos en la BBDD y sin conflictos

3.2.1.2. BAJA ALUMNOS

Descrito por Álvaro Elizalde Romero
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de baja un alumno de la academia
Actor: Gestor academia
Entrada: ID del alumno a dar de baja

Salida: ID del alumno dado de baja

Origen: GUI

Destino: BBDD

Acciones: Recibe el ID del alumno a dar de baja en la base de datos, y comprueba si es correcto, en caso de no serlo, notifica al usuario que el ID introducido no es correcto. Si es correcto, comprueba que el alumno está dado de alta en la base de datos. A continuación se pone su estado a inactivo y se notifica. En caso de no encontrarse activo se comunica que este ya ha sido dado de baja. En el caso de no encontrarse dado de alta en la base de datos se manda un mensaje.

Precondición: DNI existente en la base de datos

Postcondición: BBDD con datos correctos y sin conflictos

3.2.1.3. MODIFICAR ALUMNO

Descrito por Álvaro Elizalde Romero

Prioridad: alta

Estabilidad: alta

Descripción: modificar los datos de un alumno

Actor: gestor academia

Entrada: ID del alumno que se va a modificar

Salida: mensaje de confirmación

Origen: interfaz de alumno

Destino: sistema

Acciones: lee los datos y comprueba que los datos introducidos sean correctos y exista un alumno con el ID del alumno y esté activo. Si todo es correcto, actualiza los atributos con los datos introducidos y manda un mensaje de confirmación. En caso contrario, manda un mensaje de error.

Precondición: BBDD sin conflictos

Postcondición: no hay ni ID ni DNI de alumno repetidos en la base de datos y si éxito, alumno insertado en la base de datos

3.2.1.4. CONSULTA ALUMNO

Descrito por Álvaro Elizalde

Prioridad: alta

Estabilidad: alta

Descripción: muestra los datos de alumno

Actor: gestor academia
Entrada: DNI alumno
Salida: activo, dni, nombre, apellidos, edad, teléfono
Origen: GUI
Destino: GUI
Acciones: comprueba que el ID introducido es correcto y que está asociado a un alumno existente en la base de datos. En caso afirmativo, muestra los atributos del cliente y en caso contrario, se informa del error.
Precondición: datos de la BBDD sean correctos y no existan conflictos
Postcondición: no haya modificación de datos en la base de datos

3.2.1.5. LISTAR ALUMNOS

Descrito por Álvaro Elizalde Romero
Prioridad: alta
Estabilidad: alta
Descripción: muestra atributos de todos los alumnos registrados en la base de datos
Actor: gestor academia
Entrada: -
Salida: lista de alumnos registrados en la base de datos
Origen: GUI
Destino: GUI
Acciones: muestra todos los atributos de cada alumno registrado en la base de datos
Precondición: datos de la BBDD sean correctos y no existan conflictos
Postcondición: no haya modificación de datos en la base de datos

3.2.1.6. Query: Listar las notas de un alumno dado

Dado el DNI de un alumno y un año, se comprueba que los valores introducidos son válidos, si lo son, se busca en la base de datos que exista ese alumno y que tenga una matrícula de ese año. Si lo encuentra se listan sus notas, de lo contrario se informa de que los datos son incorrectos.

3.2.2. MÓDULO PROFESORES

3.2.2.1. ALTA PROFESORES

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de alta un nuevo profesor de la academia
Actor: Gestor academia
Entrada: DNI, Nombre Completo, Edad, Tipo (y duración del profesor a dar de alta en el caso de que el tipo seleccionado sea interino)
Salida: ID del profesor dado de alta
Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos del profesor a añadir en la base de datos, y comprueba si son correctos, en caso de no serlo notifica al usuario. Si son correctos, comprueba si este profesor ya está registrado en la base de datos, y la situación en la que se encuentra su estado.
Si el estado es false, entonces lo cambiaremos a true y informaremos de su dada de alta, sin embargo, si este es true se informa de que ya está registrado en la base de datos.
Por el contrario, si no se encuentra registrado en la base datos, y los datos insertados son correctos, este se añadirá a la base de datos con un estado igual a true, y se informará al usuario de la dada de alta del profesor.
Precondición: Que no este el DNI del profesor repetido en la base de datos y que este, junto con los demás datos, estén bien introducidos.
Postcondición: Que no haya profesores repetidos en la base de datos, si éxito, añadir dicho profesor en la base de datos.

3.2.2.2. BAJA PROFESORES

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de baja un profesor de la academia
Actor: Gestor academia

Entrada: ID del profesor a dar de baja
Salida: ID del profesor dado de baja
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del profesor a dar de baja en la base de datos, y comprueba si es correcto, en caso de no serlo, notifica al usuario. Si es correcto, comprueba si este profesor está registrado en la base de datos, y la situación en la que se encuentra su estado. Si el estado es false, entonces notificaremos al usuario de que este profesor ya había sido dado de baja, y si este es true cambiaremos el estado a false, y finalmente comunicaremos al usuario de su dada de baja.
Por el contrario si no se encuentra registrado en la base de datos, se notificará al usuario de que el profesor no se encuentra en la base de datos.
Precondición: Que no este el DNI del profesor repetido en la base de datos, es decir que no haya dos profesores iguales.
Postcondición: Que no haya profesores repetidos en la base de datos, si éxito, damos de baja al profesor de la base de datos

3.2.2.3. MODIFICAR PROFESORES

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Modifica los datos del profesor en la base de datos.
Actor: Gestor academia
Entrada: ID del profesor que se va a modificar y los campos a modificar
Salida: ID del profesor modificado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del profesor a modificar y comprueba si este se encuentra registrado en la base de datos, en caso negativo, informa al usuario de que este no se encuentra registrado, sin embargo, si este se encuentra registrado se solicitarán los datos a modificar. En el caso del campo de tipo este tendrá que ser seleccionado de forma que se mantenga el tipo inicial, en caso contrario no se realizaría la modificación y se mostraría un mensaje de error. Se comprueba si los datos

insertados son correctos, si estos no lo son se informa al usuario, en caso contrario, se procede a hacer los cambios y notificar al usuario de estos.
Precondición: Que no este el DNI del profesor repetido en la base de datos, y que este junto con los datos del profesor estén bien introducidos.
Postcondición: Que no haya profesores repetidos en la base de datos, si éxito, modificar los datos del profesor.

3.2.2.4. CONSULTA PROFESORES

Descrito por José Antonio Tortosa Aranda
Prioridad: Media
Estabilidad: Alta
Descripción: Consultar datos del profesor de la academia
Actor: Gestor academia
Entrada: ID del profesor que se va a consultar
Salida: Datos del profesor consultado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del profesor a consultar y comprueba si este se encuentra registrado en la base de datos, en caso negativo se informa al usuario de que este no se encuentra registrado, sin embargo, si este se encuentra, se mostrarán todos los datos registrados sobre este en la base de datos.
Precondición: Que no este el DNI del profesor repetido en la base de datos, es decir, que no haya dos profesores iguales.
Postcondición: Que no haya profesores repetidos en la base de datos, si éxito, mostrar datos del profesor consultado.

3.2.2.5. LISTAR PROFESORES

Descrito por José Antonio Tortosa Aranda
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra los datos de todos los profesores registrados
Actor: Gestor academia
Entrada:
Salida: Repertorio de los profesores registrados en la base de datos

Origen: Teclado
Destino: Base de datos
Acciones: Busca en la base de datos todos los datos que tenemos registrados sobre los profesores y los muestra
Precondición: Que no este el DNI del profesor repetido en la base de datos, es decir, que no haya dos profesores iguales.
Postcondición: Que no haya profesores repetidos en la base de datos, si éxito, mostrar datos de todos los profesores registrados.

3.2.2.6. LISTAR PROFESORES FIJOS

Descrito por José Antonio Tortosa Aranda
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra los datos de todos los profesores fijos registrados
Actor: Gestor academia
Entrada:
Salida: Repertorio de los profesores fijos registrados en la base de datos
Origen: Teclado
Destino: Base de datos
Acciones: Busca en la base de datos todos los datos que tenemos registrados sobre los profesores fijos y los muestra.
Precondición: Que no este el DNI del profesor repetido en la base de datos, es decir, que no haya dos profesores iguales.
Postcondición: Que no haya profesores repetidos en la base de datos, si éxito, mostrar datos de todos los profesores registrados.

3.2.2.7. LISTAR PROFESORES INTERINOS

Descrito por José Antonio Tortosa Aranda
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra los datos de todos los profesores interinos registrados
Actor: Gestor academia
Entrada:
Salida: Repertorio de los profesores interinos registrados en la base de datos
Origen: Teclado

Destino: Base de datos
Acciones: Busca en la base de datos todos los datos que tenemos registrados sobre los profesores interinos y los muestra.
Precondición: Que no este el DNI del profesor repetido en la base de datos, es decir, que no haya dos profesores iguales.
Postcondición: Que no haya profesores repetidos en la base de datos, si éxito, mostrar datos de todos los profesores registrados.

3.2.2.8. LISTAR PROFESORES POR MATRICULABLE

Descrito por Santiago González García
Prioridad: Media
Estabilidad: Alta
Descripción: Listar todos los profesores que haya en un matriculable
Actor: Gestor academia
Entrada: Identificador de matriculable
Salida: Lista de profesores
Origen: Interfaz de matriculables
Destino: Sistema
Acciones: Recibe el identificador de un matriculable a través de la interfaz, posteriormente controla si existe el matriculable en la base de datos de matriculable con el campo activo en verdadero, si se cumple se listan todos los profesores vinculados al identificador del matriculable en la base de datos de profesor-matriculable. Si todo se cumple se muestran por pantalla los datos de los profesores, si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición: No hay identificador de matriculable repetido en la base de datos ni de profesor y no hay un matriculable asociado dos veces al mismo profesor.
Postcondición: No hay identificador de matriculable repetido en la base de datos ni de profesor y no hay un matriculable asociado dos veces al mismo profesor; si éxito,se listan los profesores.

3.2.2.9. VINCULAR MATRICULABLE

Descrito por Diego Flores Simón
Prioridad: Alta
Estabilidad: Alta

Descripción: Vincula un matriculable a un profesor
Actor: Gestor academia
Entrada: Identificador de profesor y el identificador de matriculable.
Salida: Mensaje de acierto o de error
Origen: Interfaz de profesor
Destino: Sistema
Acciones: Recibe el identificador de un matriculable y uno de un profesor a través de la interfaz, posteriormente controla si existe el matriculable en la base de datos de matriculable con el campo activo en verdadero y si existe el profesor en la base de datos de profesores. Si se cumple tanto para el matriculable como para el profesor, se comprueba si no están ya vinculados en la base de datos de profesor-matriculable. Si todo se cumple se añade una nueva entrada en la base de datos de profesor-matriculable, si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición: No hay un matriculable asociado dos veces al mismo profesor.
Postcondición: No hay un matriculable asociado dos veces al mismo profesor ;si éxito, nuevo profesor-matriculable insertado en la base de datos de profesor-matriculable;

3.2.2.10. DESVINCULAR MATRICULABLE

Descrito por Diego Flores Simón
Prioridad: Alta
Estabilidad: Alta
Descripción: Desvincula un matriculable a un profesor
Actor: Gestor academia
Entrada: Identificador de profesor y el identificador de matriculable.
Salida: Mensaje de acierto o de error
Origen: Interfaz de profesor
Destino: Sistema
Acciones: Recibe el identificador de un matriculable y el identificador de un profesor a través de la interfaz, posteriormente controla si existe el matriculable en la base de datos de matriculable con el campo activo en verdadero y si existe el profesor en la base de datos de profesores y si también se encuentra activo. Se comprueba si existe la correspondiente entrada en la base de datos profesor-matriculable para comprobar si se encuentran ya vinculados. Si se cumple se elimina de la base de datos de

profesor-matriculable. En el caso de que no se encontrarán los dos IDs proporcionados como vinculados y las anteriores partes no se cumplieran se aborta la operación y se informa por pantalla.
Precondición: No hay un matriculable asociado dos veces al mismo profesor.
Postcondición: No hay un matriculable asociado dos veces al mismo profesor ;si éxito, nuevo profesor-matriculable insertado en la base de datos de profesor-matriculable.

3.2.3. MÓDULO MATRÍCULA

3.2.3.1 ABRIR MATRÍCULA

Descrito por Santiago González García
Prioridad: alta
Estabilidad: alta
Descripción: da de alta una matrícula
Actor:Gestor academia
Entrada: Identificador de alumno y descuento
Salida: Identificador de matrícula
Origen: Interfaz de matrícula
Destino: sistema
Acciones: Recibe el identificador de alumno a través de la interfaz, posteriormente controla si existe el alumno en la base de datos de alumnos con el campo activo en verdadero.Si existe el alumno y está activo se comprueba si el descuento es un valor positivo entre 0 y 1 y la matrícula no está en la base de datos , se crea una entrada en la base de datos de matrícula con el campo abierto en true y activo en true, y se devuelve el identificador de esta nueva matrícula en la base de datos de matrícula, si por el contrario ya está en la base de datos únicamente se modificaría el activo y el abierto a true ,si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición:No hay identificador de matrícula repetido en la base de datos.

Postcondición: No hay identificador de matrícula repetido en la base de datos; si éxito, nueva matrícula insertada en la base de datos;

3.2.3.2 VINCULAR MATRICULABLE

Descrito por Santiago González García

Prioridad: alta

Estabilidad: alta

Descripción: vincula un matriculable a una matrícula

Actor:gestor academia

Entrada: Identificador de matrícula, identificador de matriculable

Salida:Mensaje de acierto o de error

Origen: Interfaz de matrícula

Destino: sistema

Acciones: Recibe el identificador de un matriculable y uno de una matrícula a través de la interfaz, posteriormente controla si existe el matriculable en la base de datos de matriculable con el campo activo en true y si existe la matricula en la base de datos de matrículas y tiene el campo de abierto en true y activo en true.
Si se cumple tanto para el matriculable como para la matrícula se comprueba :si se quiere añadir se comprueban si no están ya vinculados en la base de datos de matrícula-matriculable.
Si todo se cumple se añade una nueva entrada en la base de datos de matrícula-matriculable,además de modificar el precio total aumentandolo y disminuyendo las plazas.Si alguna parte no se cumple se aborta la operación y se informa por pantalla.

Precondición: Una matrícula no se puede asociar dos veces a un matriculable

Postcondición: Una matrícula no se puede asociar dos veces a un matriculable ;si éxito, nuevo matrícula-matriculable insertado en la base de datos de matrícula-matriculable;

3.2.3.3 CERRAR MATRÍCULA

Descrito por Mario Campos Sobrino

Prioridad: alta
Estabilidad:alta
Descripción: Cambiamos el campo abierto a FALSE devolviendo el ID de la matrícula cerrada.
Actor: Gestor de academia
Entrada: ID de la matrícula a cerrar
Salida: Id de la matrícula cerrada
Origen: Interfaz de matrícula
Destino: sistema
Acciones: Recibe el identificador de la matrícula que se quiere cerrar, posteriormente comprueba si está matrícula existe en nuestra bases de datos, si no existe, informamos al usuario. De existir la matrícula, comprobamos el campo de abierto y activo pero solo cambiamos a false el abierto, indicando así al usuario que ya hemos cerrado la matrícula y devolvemos el ID de la matrícula cerrada. Si la matrícula ya estaba cerrada, informamos al usuario de que la acción que intentaba realizar ya estaba hecha.
Precondición:No hay identificador de matrícula repetido en la base de datos..
Postcondición: No hay identificador de matrícula repetido en la base de datos; si éxito se modifica el campo abierto de la matrícula a false.

3.2.3.4 BAJA MATRÍCULA

Descrito por Mario Campos Sobrino
Prioridad: alta
Estabilidad:alta
Descripción: Cambiamos el campo activo a FALSE devolviendo el ID de la matrícula cerrada.
Actor: Gestor de academia
Entrada: ID de la matrícula a cerrar
Salida: Id de la matrícula cerrada
Origen: Interfaz de matrícula
Destino: sistema
Acciones: Recibe el identificador de la matrícula que se quiere cerrar, posteriormente comprueba si está matrícula existe en nuestra bases de datos, si no existe, informamos al usuario. De existir la matrícula, comprobamos el campo de abierto está a false y activo a true, cambiamos a false el activo, indicando así al usuario que ya

hemos dado de baja la matrícula y devolvemos el ID de la matrícula dada de baja. Si la matrícula ya estaba dada de baja, informamos al usuario de que la acción que intentaba realizar ya estaba hecha.
Precondición:No hay identificador de matrícula repetido en la base de datos..
Postcondición: No hay identificador de matrícula repetido en la base de datos; si éxito se modifica el campo activo de la matrícula a false.

3.2.3.5 MODIFICACIÓN MATRÍCULA

Descrito por Santiago González García
Prioridad: media
Estabilidad:alta
Descripción: Modifica una matrícula
Actor: gestor academia
Entrada: Identificador de matrícula, descuento
Salida: Mensaje de acierto o de error.
Origen: Interfaz de matrícula
Destino: sistema
Acciones: Recibe el identificador de la matrícula a través de la interfaz, posteriormente controla si existe la matrícula en la base de datos de matrícula y tiene el campo abierto en false pero activo a true.Si existe la matricula se modifica la entrada en la base de datos de la matrícula y se indica por pantalla que la operación ha sido exitosa,si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición:No hay identificador de matrícula repetido en la base de datos.
Postcondición: No hay identificador de matrícula repetido en la base de datos; si éxito, matrícula modificada en la base de datos;

3.2.3.6 CONSULTA MATRÍCULA

Descrito por Santiago González García
Prioridad: media

Estabilidad:alta
Descripción: Consulta la información de la matrícula
Actor:gestor academia
Entrada: Identificador de matrícula
Salida: Información de la matrícula o mensaje de error.
Origen: Interfaz de matrícula
Destino: sistema
Acciones: Recibe el identificador de la matrícula a través de la interfaz, posteriormente controla si existe la matrícula en la base de datos de matrícula y tiene el campo abierto en false y el activo a true.
Si existe la matrícula se genera un TOA con toda la información de esta accediendo a la base de datos de alumno, a través del identificador de alumno, y se accede a todos los matriculables de la matrícula accediendo a sus identificadores en la base de datos de matrícula-matriculable , posteriormente se accede a su información en la base de datos de matriculable.
Una vez formado el TOA se saca por pantalla toda la información ,si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición:Una matrícula no se puede asociar dos veces a un matriculable y no hay identificadores de matrícula repetidos
Postcondición: Una matrícula no se puede asociar dos veces a un matriculable y no hay identificadores de matrícula repetidos ;si éxito, muestran por pantalla los datos de la matrícula.

3.2.3.7 LISTAR MATRÍCULAS

Descrito por Santiago Santé Pena
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra una lista de todas las matrículas en la base de datos activas o no
Actor: Gestor Academia
Entrada:
Salida: Repertorio de las matrículas existentes en la base de datos
Origen: Teclado
Destino: Base de datos

Acciones: Realiza una búsqueda de todas las matrículas y generará una lista que será mostrada por pantalla
Precondición: No hay identificador de matrícula repetido
Postcondición: Si no hay identificadores de matrícula repetidos mostrar por pantalla

3.2.3.8 LISTAR MATRICULABLES DE MATRÍCULA

Descrito por Santiago González García
Prioridad: media
Estabilidad: alta
Descripción: Muestra todos los matriculables asociados a una matrícula
Actor: Gestor de academia
Entrada: Identificador de matrícula
Salida:Lista de matriculables
Origen: Interfaz matrícula
Destino: GUI
Acciones: Recibe el identificador del matriculable a través de la interfaz, posteriormente controla si existe el matriculable en la base de datos de matriculable.Si existe la matriculable se accede a la base de datos de matrícula-matriculable y se muestra por pantalla los identificadores de los matriculables junto a la nota que les corresponde, si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición:No hay identificador de matriculable repetido en la base de datos ni de matriculable y no hay un matriculable asociado dos veces a la misma matrícula .
Postcondición: No hay identificador de matrícula repetido en la base de datos ni de matriculable y no hay un matriculable asociado dos veces a la misma matrícula; si éxito,se muestran los matriculables.

3.2.3.9 CONSULTAR NOTA POR MATRICULABLE

Descrito por Santiago González García
Prioridad: media
Estabilidad: alta
Descripción: Muestra la nota de un matriculable dado la matrícula
Actor: Gestor de academia
Entrada: Identificador de matrícula y identificador de matriculable

Salida:Lista de matriculables
Origen: Interfaz matrícula
Destino: GUI
Acciones: Recibe el identificador de la matrícula y del matriculable a través de la interfaz, posteriormente controla si existe la matrícula en la base de datos de matrícula con el campo abierto en false, y si existe el matriculable en la base de datos de matrícula con el campo activo en verdadero .Si existe la matrícula y el matriculable y están activas se accede a la base de datos de matrícula-matriculable para comprobar si están asociados, si lo están y el campo no es nulo, se muestra por pantalla la nota , si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición:No hay identificador de matrícula repetido en la base de datos ni de matriculable y no hay un matriculable asociado dos veces a la misma matrícula.
Postcondición: No hay identificador de matrícula repetido en la base de datos ni de matriculable y no hay un matriculable asociado dos veces a la misma matrícula; si éxito,se muestra la nota.

3.2.3.10 LISTAR MATRÍCULAS POR ALUMNO

Descrito por Santiago González García
Prioridad: media
Estabilidad: alta
Descripción: muestra todas las matrículas asociadas a un alumno
Actor: gestor academia
Entrada: DNI alumno
Salida: lista de matrículas asociadas a alumno
Origen: Interfaz Matrícula
Destino: GUI
Acciones: Recibe el DNI de un alumno a través de la interfaz, posteriormente controla si existe la un alumno con ese DNI en la base de datos de alumno con el campo activo en verdadero. Si existe el alumno y está activo se accede a la base de datos de matrícula para comprobar si tiene matrículas asociadas asociadas, si lo están se muestran por pantalla las matrículas con el campo abierto en false, si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición: DNI no se repite en la base de datos de alumno y id no se repite en la base de datos de matrícula.

Postcondición: DNI no se repite en la base de datos de alumno y id no se repite en la base de datos de matrícula; si éxito se muestran por pantalla las matrículas.

3.2.3.11 CALIFICAR MATRICULABLE DE MATRICULA

Autor: Santiago González García
Prioridad: media
Estabilidad: alta
Descripción: Califica la nota de un matriculable dado la matrícula
Actor: Gestor de academia
Entrada: Identificador de matrícula, identificador de matriculable y nota
Salida:Lista de matriculables
Origen: Interfaz matrícula
Destino: GUI
Acciones: Recibe el identificador de la matrícula y del matriculable a través de interfaz, posteriormente controla si existe la matrícula en la base de datos de matrícula y si existe el matriculable en la base de datos de matriculable con el campo activo en verdadero .Si existe la matrícula y el matriculable y matriculable y matrícula están activo y la matrícula tiene el atributo abierto en false y la nota introducida es un número entre el 0 y el 10, se accede a la base de datos de matrícula-matriculable para comprobar si están asociados, si lo están se sustituye el campo nota y se confirma por pantalla , si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición:No hay identificador de matrícula repetido en la base de datos ni de matriculable y no hay un matriculable asociado dos veces a la misma matrícula
Postcondición: No hay identificador de matrícula repetido en la base de datos ni de matriculable y no hay un matriculable asociado dos veces a la misma matrícula; si éxito,se califica el matriculable.

3.2.3.12 DESVINCULAR MATRICULABLE

Descrito por Santiago González García
Prioridad: alta
Estabilidad: alta
Descripción: desvincula un matriculable a una matrícula
Actor:gestor academia

Entrada: Identificador de matrícula, identificador de matriculable
Salida:Mensaje de acierto o de error
Origen: Interfaz de matrícula
Destino: sistema
Acciones: Recibe el identificador de un matriculable y uno de una matrícula a través de la interfaz, posteriormente controla si existe el matriculable en la base de datos de matriculable con el campo activo en true y si existe la matricula en la base de datos de matrículas y tiene el campo de abierto en true y activo en true.
Si se cumple tanto para el matriculable como para la matrícula se comprueba si estan vinculados.Si todo se cumple se elimina la entrada en la base de datos de matrícula-matriculable,además de modificar el precio total disminuyendolo y aumentando las plazas.Si alguna parte no se cumple se aborta la operación y se informa por pantalla.
Precondición:Una matrícula no se puede asociar dos veces a un matriculable
Postcondición: Una matrícula no se puede asociar dos veces a un matriculable ;si éxito, nuevo matrícula-matriculable insertado en la base de datos de matrícula-matriculable;

3.2.4. MÓDULO ASIGNATURAS

3.2.4.1. ALTA ASIGNATURA

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de alta una asignatura de la academia
Actor: Gestor academia
Entrada: Nombre, Tipo, Itinerario y Nivel de la asignatura a dar de alta
Salida: ID de la asignatura dada de alta

Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos de la asignatura a añadir en la base de datos, y comprueba si son correctos, en caso de no serlo notifica al usuario. Si son correctos, comprueba si esta asignatura ya está registrada en la base de datos, y la situación en la que se encuentra su estado. Si el estado es false, entonces lo cambiaremos a true y informaremos de su dada de alta, sin embargo, si este es true se informa de que ya está registrada en la base de datos.
Por el contrario, si no se encuentra registrada en la base datos, y los datos insertados son correctos, esta se añadirá a la base de datos con un estado igual a true, y se informará al usuario de la dada de alta de la asignatura.
Precondición: Que no este el Id de la asignatura repetida en la base de datos y que este, junto con el Nombre, Tipo, Itinerario y Nivel, estén bien introducidos.
Postcondición: Que no haya asignatura repetidas en la base de datos, si éxito, añadir dicha asignatura en la base de datos.

3.2.4.2. BAJA ASIGNATURA

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de baja una asignatura de la academia
Actor: Gestor academia
Entrada: ID de la asignatura a dar de baja
Salida: ID de la asignatura dada de baja
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID de la asignatura a dar de baja en la base de datos, y comprueba si es correcto, en caso de no serlo, notifica al usuario que el ID introducido no es correcto. Si es correcto, comprueba si esta asignatura está registrada en la base de datos, y la situación en la que se encuentra su estado.
Si el estado es false, entonces notificaremos al usuario de que esta asignatura ya había sido dado de baja, y si este es true, cambiaremos el estado de este a false, y finalmente comunicaremos al usuario su dada de baja.

Por el contrario si no se encuentra registrada en la base de datos, se notificará al usuario de que la asignatura no se encuentra en la base de datos.
Precondición: Que no este el ID de la asignatura repetida en la base de datos, es decir, que no haya dos asignaturas iguales.
Postcondición: Que no haya asignaturas repetidas en la base de datos, si éxito, damos de baja la asignatura de la base de datos.

3.2.4.3. MODIFICACIÓN ASIGNATURA

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Modifica los datos de la asignatura en la base de datos.
Actor: Gestor academia
Entrada: ID de la asignatura que se va a modificar
Salida: ID del asignatura modificado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID de la asignatura a modificar y comprueba si es correcto, en caso de no serlo, notifica al usuario que el ID introducido no es correcto.
Si es correcto, comprueba si esta asignatura se encuentra registrada en la base de datos, en caso negativo se informa al usuario de que esta no se encuentra registrada, sin embargo, si esta se encuentra registrada se solicitarán los datos a modificar.
Se comprueba si los datos insertados son correctos, si estos no lo son se informa al usuario, en caso contrario, se procede a hacer los cambios y notificar al usuario de estos.
Precondición: Que no este el ID de la asignatura repetido en la base de datos, y que este junto con los datos de la asignatura estén bien introducidos.
Postcondición: Que no haya asignaturas repetidas en la base de datos, si éxito, modificar los datos de la asignatura.

3.2.4.4. CONSULTA ASIGNATURA

Descrito por José Antonio Tortosa Aranda

Prioridad: Media
Estabilidad: Alta
Descripción: Consultar datos de la asignatura de la academia
Actor: Gestor academia
Entrada: ID de la asignatura que se va a consultar
Salida: Datos de la asignatura consultada
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID de la asignatura a consultar y comprueba si es correcto, en caso de no serlo, notifica al usuario que el ID introducido no es correcto.
Si es correcto, comprueba si esta se encuentra registrada en la base de datos, en caso negativo, se informa al usuario de que esta no se encuentra registrada, sin embargo, si este se encuentra registrada se mostrarán todos los datos registrados sobre esta asignatura en la base de datos.
Precondición: Que no este el ID de la asignatura repetida en la base de datos, es decir, que no haya dos asignaturas iguales.
Postcondición: Que no haya asignaturas repetidas en la base de datos, si éxito, mostrar datos de la asignatura consultada.

3.2.4.5. LISTAR MATRICULABLES POR ASIGNATURA

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Muestra los datos de todos los matriculables por asignatura
Actor: Gestor academia
Entrada: ID de la asignatura
Salida: Datos de los matriculables que comparten ese mismo ID
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID de la asignatura, comprueba si está bien introducida, en caso contrario, informa al usuario de esto. Sin embargo, si está bien introducida comprueba si existe dicha asignatura en la base de datos, en caso afirmativo, muestra los matriculables relacionados a esta, en caso negativo, informa al usuario de que la asignatura no se encuentra registrada en la base de datos.

Precondición: Que no este el ID de la asignatura repetida en la base de datos, es decir que no haya dos asignaturas iguales.
Postcondición: Que no haya asignaturas repetidas en la base de datos, si éxito, mostrar todos los datos de matriculables registrados en la base de datos de la asignatura correspondiente.

3.2.4.6. QUERY: NOTA MEDIA DE UNA ASIGNATURA DADA

Dado el ID de una asignatura y un año, se comprueba que los valores introducidos son válidos, si lo son, se busca en la base de datos que exista esa asignatura y que tenga un matriculable de ese año. Si lo encuentra se lista una media de las notas de todos los alumnos matriculados en ese año, pueden ser alumnos de diferentes matriculables, de lo contrario se informa de que los datos son incorrectos.

3.2.5. MÓDULO AÑO

3.2.5.1. ALTA AÑO

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de alta un año
Actor: Gestor academia
Entrada: Año
Salida: ID de año registrado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el año a registrar en la base de datos, comprueba si el año está bien introducido, en caso negativo, informa al usuario que el año introducido no es correcto, en caso afirmativo, comprueba si está ya registrado en la base de datos, en ese caso, informa al usuario que ya se encuentra en la base de datos, en caso

contrario, lo añade y informa al usuario de que se ha dado de alta el año correctamente.
Precondición: Que no este el ID del año repetido en la base de datos y que este, esté introducido correctamente.
Postcondición: Que no haya años repetidos en la base de datos, si éxito, añadir dicho año en la base de datos.

3.2.5.2. BAJA AÑO

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de baja un año de la academia
Actor: Gestor academia
Entrada: ID del año a dar de baja
Salida: ID del año dado de baja
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del año a dar de baja, comprueba si el ID está bien introducido, en caso negativo, informa al usuario de que el ID no está bien introducido, en caso afirmativo, comprueba si el año está registrado en la base de datos. Si no lo está, informa al usuario de que ese año no se encuentra registrado, si lo está, comprueba cómo se encuentra el estado. Si este es true, lo convierte a false, he informa al usuario que se ha dado de baja el año correctamente, si está a false, no se realiza ningún cambio y se informa al usuario que ese año ya estaba dado de baja.
Precondición: Que no este el ID del año repetido en la base de datos, es decir, que no haya dos años iguales.
Postcondición: Que no haya años repetidos en la base de datos, si éxito, damos de baja al año de la base datos

3.2.5.3. MODIFICACIÓN AÑO

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Modifica los datos del año en la base de datos.

Actor: Gestor academia
Entrada: ID del año que se va a modificar
Salida: ID del año modificado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del año a modificar y comprueba si es correcto, en caso de no serlo, notifica al usuario que el ID introducido no es correcto.
Si es correcto, comprueba si este año se encuentra registrada en la base de datos, en caso negativo se informa al usuario de que este no se encuentra registrado, sin embargo, si este se encuentra registrado se solicitarán los datos a modificar.
Se comprueba si los datos insertados son correctos, si estos no lo son, se informa al usuario, en caso contrario, se procede a hacer los cambios y notificar al usuario de estos.
Precondición: Que no este el ID del año repetido en la base de datos, y que este junto con los datos del año estén bien introducidos.
Postcondición: Que no haya años repetidos en la base de datos, si éxito, modificar los datos del año.

3.2.5.4. CONSULTA AÑO

Descrito por José Antonio Tortosa Aranda
Prioridad: Media
Estabilidad: Alta
Descripción: Consultar datos del año de la academia
Actor: Gestor academia
Entrada: ID del año que se va a consultar
Salida: Datos del año consultado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del año a consultar, comprueba si el ID introducido es correcto, en caso negativo, informa al usuario que el ID introducido no es correcto, en caso afirmativo, comprueba si este se encuentra registrado en la base de datos, si no se encuentra, se informa al usuario de que este no está, sin embargo, si este se encuentra registrado se mostrarán todos los datos registrados sobre este en la base de datos.

Precondición: Que no este el ID del año repetido en la base de datos, es decir que no haya dos años iguales.
Postcondición: Que no haya años repetidos en la base de datos, si éxito, mostrar datos del año consultado.

3.2.5.5. LISTAR AÑO

Descrito por Diego Flores Simón
Prioridad: Alta
Estabilidad: Alta
Descripción: Acceder a todos los años que tenemos en nuestra base de datos.
Actor: Gestor academia
Entrada: —
Salida: Todas los años que tenemos en nuestra base de datos con sus datos correspondientes
Origen: Teclado
Destino: Base de datos
Acciones: Busca en la base de datos todos los años que tenemos y los muestra e informa de que se ha realizado bien la función o que no tengamos ningún año y se informe de ello
Precondición: Que no haya años repetidos en la base de datos, es decir, que el ID y el año no estén repetidos.
Poscondición: Que no haya años repetidos en la base de datos, si éxito, muestran todos en una lista.

3.2.5.6. Query: Cuántos alumnos hay matriculados dado un año un rango de años

Dado un año, se comprueba que los valores introducidos son válidos, si lo son, se busca en la base de datos que existan matriculables en ese año o entre esos años. Si lo encuentra se lista el número de alumnos matriculados en ese año, incluye a todos los alumnos de todos los matriculables, de lo contrario se informa de que los datos son incorrectos.

3.2.6. MÓDULO GRUPO

3.2.6.1. ALTA GRUPO

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de alta un grupo
Actor: Gestor academia
Entrada: grupo
Salida: ID de grupo registrado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el grupo a registrar en la base de datos, comprueba si el grupo está bien introducido, en caso negativo, informa al usuario que el grupo introducido no es correcto, en caso afirmativo, comprueba si está ya registrado en la base de datos, en ese caso, informa al usuario que ya se encuentra en la base de datos, en caso contrario, lo añade y informa al usuario de que se ha dado de alta el grupo correctamente.
Precondición: Que no este el ID del grupo repetido en la base de datos y que este, esté introducido correctamente.
Postcondición: Que no haya grupos repetidos en la base de datos, si éxito, añadir dicho grupo en la base de datos.

3.2.6.2. BAJA GRUPO

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Dar de baja un grupo de la academia
Actor: Gestor academia
Entrada: ID del grupo a dar de baja

Salida: ID del grupo dado de baja
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del grupo a dar de baja, comprueba si el ID está bien introducido, en caso negativo, informa al usuario de que el ID no está bien introducido, en caso afirmativo, comprueba si el grupo está registrado en la base de datos. Si no lo está, informa al usuario de que ese grupo no se encuentra registrado, si lo está, comprueba cómo se encuentra el estado. Si este es true, lo convierte a false, he informa al usuario que se ha dado de baja el grupo correctamente, si está a false, no se realiza ningún cambio y se informa al usuario que ese grupo ya estaba dado de baja.
Precondición: Que no este el ID del grupo repetido en la base de datos, es decir, que no haya dos grupos iguales.
Postcondición: Que no haya grupos repetidos en la base de datos, si éxito, damos de baja al grupo de la base datos

3.2.6.3. MODIFICACIÓN GRUPO

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Modifica los datos del grupo en la base de datos.
Actor: Gestor academia
Entrada: ID del grupo que se va a modificar
Salida: ID del grupo modificado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del grupo a modificar y comprueba si es correcto, en caso de no serlo, notifica al usuario que el ID introducido no es correcto.
Si es correcto, comprueba si este grupo se encuentra registrado en la base de datos, en caso negativo se informa al usuario de que este no se encuentra registrado, sin embargo, si este se encuentra registrado se solicitarán los datos a modificar.
Se comprueba si los datos insertados son correctos, si estos no lo son, se informa al usuario, en caso contrario, se procede a hacer los cambios y notificar al usuario de estos.

Precondición: Que no este el ID del grupo repetido en la base de datos, y que este junto con los datos del grupo estén bien introducidos.
Postcondición: Que no haya grupos repetidos en la base de datos, si éxito, modificar los datos del grupo.

3.2.6.4. CONSULTA GRUPO

Descrito por José Antonio Tortosa Aranda
Prioridad: Media
Estabilidad: Alta
Descripción: Consultar datos del grupo de la academia
Actor: Gestor academia
Entrada: ID del grupo que se va a consultar
Salida: Datos del grupo consultado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del grupo a consultar, comprueba si el ID introducido es correcto, en caso negativo, informa al usuario que el ID introducido no es correcto, en caso afirmativo, comprueba si este se encuentra registrado en la base de datos, si no se encuentra, se informa al usuario de que este no está, sin embargo, si este se encuentra registrado se mostrarán todos los datos registrados sobre este en la base de datos.
Precondición: Que no este el ID del grupo repetido en la base de datos, es decir que no haya dos grupos iguales.
Postcondición: Que no haya grupos repetidos en la base de datos, si éxito, mostrar datos del grupo consultado.

3.2.6.5. LISTAR GRUPO

Descrito por Diego Flores Simón
Prioridad: Alta
Estabilidad: Alta
Descripción: Acceder a todos los grupos que tenemos en nuestra base de datos.
Actor: Gestor academia
Entrada: —
Salida: Todos los grupos que tenemos en nuestra base de datos con sus datos correspondientes

Origen: Teclado
Destino: Base de datos
Acciones: Busca en la base de datos todos los grupos que tenemos y los muestra e informa de que se ha realizado bien la función o que no tengamos ningún grupo y se informe de ello
Precondición: Que no haya grupos repetidos en la base de datos, es decir, que el ID y el año no estén repetidos.
Poscondición: Que no haya grupos repetidos en la base de datos, si éxito, muestran todos en una lista.

3.2.6.6. LISTAR MATRICULABLES POR GRUPO

Descrito por José Antonio Tortosa Aranda
Prioridad: Alta
Estabilidad: Alta
Descripción: Muestra los datos de todos los matriculables por grupo
Actor: Gestor academia
Entrada: ID de la grupo
Salida: Datos de los matriculables que comparten ese mismo ID
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el ID del grupo, comprueba si está bien introducido, en caso contrario, informa al usuario de esto. Sin embargo, si está bien introducido comprueba si existe dicho grupo en la base de datos, en caso afirmativo, muestra los matriculables relacionados a este, en caso negativo, informa al usuario de que el grupo no se encuentra registrado en la base de datos
Precondición: Que no este el ID del grupo repetido en la base de datos, es decir, que no haya dos grupos iguales.
Postcondición: Que no haya grupos repetidas en la base de datos, si éxito, mostrar todos los datos de matriculables registrados en la base de datos del grupo correspondiente.

3.2.6.7. MÓDULO MATRICULABLE

3.2.6.8. ALTA MATRICULABLE

Descrito por Mario Campos Sobrino

Prioridad: Alta

Estabilidad: Alta

Descripción: Dar de alta un matriculable

Actor: Gestor academia

Entrada:Hora, ID de Asignatura, ID de Grupo, ID de Año, Precio

Salida: ID del matriculable

Origen: Teclado

Destino: Base de datos

Acciones: Recibe los datos del matriculable a añadir en la base de datos, y comprueba si son correctos, en caso de no serlo notifica al usuario. Si son correctos, comprueba si este matriculable ya está registrado en la base de datos, y la situación en la que se encuentra su estado. Si el estado es false, entonces lo cambiaremos a true e informaremos de su dada de alta, sin embargo si este es true se informa de que ya está registrado en la basé de datos.

Por el contrario si no se encuentra registrado en la base datos, y los datos insertados son correctos, este se añadirá a esta con un estado igual a true, y se informará al usuario de la dada del alta del matriculable..

Precondición: Que no este el Id del matriculable repetido en la base de datos y que todos sus datos, tanto la Hora, ID de Asignatura, ID de Grupo, ID de Año y Precio, están bien introducidos.

Postcondición: Que no haya matriculables repetidos en la base de datos, si éxito, añadir dicha matriculable en la base de datos.

3.2.6.9. BAJA MATRICULABLE

Descrito por Mario Campos Sobrino

Prioridad: Alta

Estabilidad: Alta

Descripción: Dar de baja un matriculable

Actor: Gestor academia

Entrada: ID del matriculable

Salida: ID del matriculable

Origen: Teclado
Destino: Base de datos
Acciones: Recibe los datos del matriculable a dar de baja en la base de datos, y comprueba si son correctos, en caso de no serlo notifica al usuario. Si son correctos, comprueba si este matriculable está registrado en la base de datos, y la situación en la que se encuentra su estado. Si el estado es false, entonces notificaremos al usuario de que este matriculable ya había sido dado de baja, y si este es true cambiaremos el estado de este a false, y finalmente comunicaremos al usuario su dada de baja.
Por el contrario si no se encuentra registrado en la base de datos, se notificará al usuario de que el matriculable no se encuentra en la base de datos.
Precondición: Que no este el ID del matriculable repetido en la base de datos, es decir que no haya dos matriculables iguales.
Postcondición: Que no haya matriculables repetidos en la base de datos, si éxito, eliminamos el matriculable de la base de datos

3.2.6.10. MODIFICACIÓN MATRICULABLE

Descrito por Mario Campos Sobrino
Prioridad: Alta
Estabilidad: Alta
Descripción: Modificar un matriculable y sus datos
Actor: Gestor academia
Entrada: ID del matriculable que se va a modificar
Salida: ID del matriculable modificado
Origen: Teclado
Destino: Base de datos
Acciones: Recibe el id del matriculable a modificar, y comprueba si este se encuentra registrado en la base de datos, en caso negativo se informa al usuario de que este no se encuentra registrado, sin embargo, si esta se encuentra registrado se solicitarán los datos a modificar. Se comprueba si los datos insertados son correctos, si estos no lo son se informa al usuario, y en caso afirmativo se procede a hacer los cambios en la base de datos y notificar al usuario de estos.
Precondición: Que no este el ID del matriculable repetido en la base de datos, y que este junto con los datos del matriculable estén bien introducidos.
Postcondición: Que no haya matriculables repetidos en la base de datos, si éxito,

3.2.6.11. CONSULTAR MATRICULABLE

Descrito por Mario Campos Sobrino

Prioridad: Alta

Estabilidad: Alta

Descripción: Poder ver y consultar los datos de un matriculable

Actor: Gestor academia

Entrada: ID del matriculable que se va a consultar

Salida: Datos del matriculable consultado

Origen: Teclado

Destino: Base de datos

Acciones: Recibe el id del matriculable a consultar, y comprueba si este se encuentra registrado en la base de datos, en caso negativo se informa al usuario de que esta no se encuentra registrado, sin embargo, si este se encuentra registrado se mostrarán todos los datos registrados sobre este en la base de datos.

Precondición: Que no este el ID del matriculable repetido en la base de datos, es decir que no haya dos matriculables iguales.

Postcondición: Que no haya matriculables repetidos en la base de datos, si éxito, mostrar datos del matriculable consultado.

3.2.6.12. LISTAR MATRICULABLE

Descrito por Mario Campos Sobrino

Prioridad: Alta

Estabilidad: Alta

Descripción: Acceder a todos los matriculables que tenemos en nuestra base de datos.

Actor: Gestor academia

Entrada: —

Salida: Todas los matriculables que tenemos en nuestra base de datos con sus datos correspondientes

Origen: Teclado

Destino: Base de datos

Acciones: Busca en la base de datos todos los matriculables que tenemos y los muestra e informa de que se ha realizado bien la función o que no tengamos ningún matriculable y se informe de ello
Precondición: Que no haya matriculables repetidos en la base de datos, es decir, que el ID no esté repetido.
Poscondición: Que no haya matriculables repetidos en la base de datos, si éxito, muestran todos en una lista.

3.2.6.13. LISTAR MATRICULABLES POR AÑO

Descrito por Mario Campos Sobrino
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra los datos de todos los matriculables por año
Actor: Gestor academia
Entrada: ID del año
Salida: Datos de los matriculables que comparten ese mismo ID
Origen: Teclado
Destino: Base de datos
Acciones: Recibe ID de año, comprueba si este año existe en la BBDD. Si existe, muestra los matriculables relacionados a este año, donde el idAño del matriculable sea el mismo al ID introducido. Si no existe, informará al usuario de que este año no existe en nuestra Base de Datos.
Precondición: Que no este el ID del año repetido en la base de datos, es decir, que no haya dos años iguales.
Postcondición: Que no haya años repetidos en la base de datos, si éxito, mostrar todos los datos de matriculables registrados en la base de datos del año correspondiente.

3.2.6.14. LISTAR MATRICULABLES POR GRUPO

Descrito por Mario Campos Sobrino
Prioridad: Media
Estabilidad: Alta
Descripción: Muestra los datos de todos los matriculables por un grupo
Actor: Gestor academia
Entrada: ID del grupo

Salida: Datos de los matriculables que comparten ese mismo ID
Origen: Teclado
Destino: Base de datos
Acciones: Recibe ID de grupo, comprueba si este grupo existe en la BBDD. Si existe, muestra los matriculables relacionados a este grupo, donde el idGrupo del matriculable sea el mismo al ID introducido. Si no existe, informará al usuario de que este grupo no existe en nuestra Base de Datos.
Precondición: Que no este el ID del grupo repetido en la base de datos, es decir, que no haya dos grupos iguales.
Postcondición: Que no haya grupos repetidos en la base de datos, si éxito, mostrar todos los datos de matriculables registrados en la base de datos del grupo correspondiente.

3.2.6.15. LISTAR MATRICULABLES POR ASIGNATURA

Descrito por Álvaro Elizalde Romero
Prioridad: media
Estabilidad: alta
Descripción: muestra todos los matriculables asociados a una asignatura
Actor: gestor academia
Entrada: id asignatura
Salida: lista de matriculables
Origen: teclado
Destino: base de datos
Acciones: recibe ID de asignatura y comprueba que está en la base de datos. En caso positivo, muestra todos los matriculables asociados a la asignatura, y en caso negativo, muestra un mensaje de error.
Precondición : base de datos sin datos repetidos
Postcondición: base de datos sin modificar

3.2.6.16. LISTAR MATRICULABLES POR PROFESOR

Descrito por Álvaro Elizalde Romero
Prioridad: media
Estabilidad: alta
Descripción: muestra todos los matriculables asociados a una asignatura
Actor: gestor academia

Entrada: DNI profesor

Salida: lista de matriculables

Origen: teclado

Destino: base de datos

Acciones: recibe el DNI del profesor y comprueba que está en la base de datos. En caso positivo, muestra todos los matriculables asociados al profesor, y en caso negativo, muestra un mensaje de error.

Precondición : base de datos sin datos repetidos

Postcondición: base de datos sin modificar

3.3. Requisitos de rendimiento

Número de terminales: 1
Número de usuarios: 1
Número de transacciones por segundo: Indefinido
Ordenador y memoria estándar

3.4. Modelo de dominio