**MINUTA**

Minuta No. 005 MGASoluciones

Fecha : Lunes 12 del febrero

Hora : 1:10 am

Lugar : Aula 07, UCR Sede del Atlántico

Siendo las 01:10 am horas del día 12 de febrero de 2016, se reúne el grupo: **MGASoluciones**, en sesión extraordinaria previa citación realizada.

**ASISTENTES:**

La persona encargada que precede la reunión, Alberth Calderón Alvarado, pasó lista y determinó que había quórum.

Los asistentes a esta reunión son:

· Albert Calderón Alvarado. \_\_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_

· Anthony Hernández Badilla \_\_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_

· Gustavo Nájera Nájera. \_\_ \_Presente\_\_\_\_\_\_\_\_\_

· Michael Meléndez Mesén. \_\_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_

**AGENDA:**

1. Se asigna tareas Individuales a Alberth Calderon Alvarado.

**ASUNTOS:**

1. **Asuntos tratados y acuerdos tomados**

* Compra de artículos
* Categorías de artículos
* Proveedores
* Un web servicie para que el proveedor reciba el pedido.

1. **Tareas concluidas con éxito.** 
   1. **Capa data: Proyecto Inventario**
      1. supplierData:
         1. **insertSupplier($supplier):** Método para insertar un nuevo proveedor en la base de datos, controla que no exista en la base, y le asigna un nuevo número de identificación consecutivo.
         2. **updateSupplier($supplier):** Actualiza los datos de un proveedor en la base de datos.
         3. **desactiveSupplier($supplier):** Desactiva un Proveedor en la base de datos
         4. **activeSupplier():** Activa un Proveedor en la base de datos
         5. **getAllSupplierDesactive():** Obtiene un array de todos los proveedores inactivos
         6. **getAllSupplierActive():**Obtiene un array de todos los proveedores activos
         7. **getAllSupplier():**Obtiene un array de todos los proveedores.
      2. stockData
         1. **insertStock($stock);** registra un artículo en inventario
         2. **stockExist($idProduct,$idStore):** comprueba si el archivo ya existe registrado en la base.
         3. **updateStock($stock):** actualiza un registro del inventario
         4. **getAllStock():** obtiene todos los registros almacenados
         5. **insertExist():** método que obtiene la lista de productos existentes y comprueba si existen registrados en inventario, si el producto no está registrado inserta un nuevo dato de inventario con los valores de cantidad en cero
         6. **getNameProduct($idProduct):** obtiene el nombre del producto apartar de su id que posee el dato de inventario
         7. **getBrandProduct($idProduct);** obtiene la marca del producto a partir de su id que posee el dato de inventario.
         8. **getModelProduct($idProduct):** obtiene el modelo del producto a partir de su id que posee el dato de inventario
         9. **getNameStore($idProduct**): obtiene el nombre dela tienda a partir de su id que posee el dato de inventario
      3. purchaseData:
         1. **getAllPurchase():** Obtiene todos los registros
         2. **getAllPurchaseSupplier($idSupplier):** obtiene el histórico de compras por proveedor.
   2. **Capa data: webservice**
      1. purchaseData:
         1. **insertPurchase($purchase):** Inserta en la base de datos el registro de un nuevo pedido realizado por la tienda
         2. **getNameProduct($idProduct):** Obtiene el nombre del producto apartar de su id que posee el dato de inventario
         3. **getBrandProduct($idProduct);** Obtiene la marca del producto a partir de su id que posee el dato de inventario.
         4. **getModelProduct($idProduct):** Obtiene el modelo del producto a partir de su id que posee el dato de inventario
         5. **getNameSupplier($idSupplier):** Obtiene el nombre del proveedor atreves de su id.
   3. **Capa Domain Inventario**
      1. **Purchase:** $idPurchases, $idSupplier, $datePurchase, $descriptionPurchase, $idProduct, $totalPurchase, sets y gets
      2. **Stock:** $idStock, $idProduct, $idStore, $idQuantity, $levelStock, sets y gets
      3. **Supplier:** $idSupplier, $nameSupplier, $emailSupplier, $telephoneSupplier, sets y gets
   4. **Capa Domain web service**
      1. **Purchase:** $idPurchases, $idSupplier, $datePurchase, $descriptionPurchase, $idProduct, $totalPurchase, sets y gets
      2. **Stock:** $idStock, $idProduct, $idStore, $idQuantity, $levelStock, sets y gets
      3. **Supplier:** $idSupplier, $nameSupplier, $emailSupplier, $telephoneSupplier, sets y gets
2. **Capa Data:**

Se creó la clase AccountData, la cual funciona como enlace de datos, es decir proporciona las funciones que realizan las consultas a la Base de datos.

Las funciones que incluye la presente clase son:

1. getIDData()

El método funciona como medio para obtener un nuevo id, mediante la obtención del id más alto registrado en la tabla tbAccount en la Base de datos y sumándole uno a ese valor, por ende el retorno seria de tipo numérico.

1. getAllAccountAssetsData()

El presente método obtiene CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount; de todas las cuentas activas registradas en la base de datos.

1. getAccountByIdData($idAccount)

Recibe por parámetros un valor numérico, el cual se atribuye al id de la cuenta de la cual se desea obtener CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

1. insertAccountData($account)

Recibe por parámetros un objeto de tipo Account que corresponde a la clase Account mencionada antes en el presente documento.

La función del método corresponde a la inserción de una cuenta en la tabla Account en la Base de Datos.

1. DeleteAccountData($idAccount)

Recibe por parametros un valor numérico que corresponde al id de una cuenta registrada previamente en la Base de datos.

El objetivo del método deleteAccountData es eliminar la cuenta que corresponde al id recibido por parámetros.

1. updateAccountData($account)

Recibe por parametros un objeto de tipo Account.

El objetivo del método updateAccountData es buscar coincidencias mediante la comparación entre el id de las cuentas en la Base de Datos y el id de la cuenta recibida por parametros, en caso de encontrar coincidencia se actualizan los datos de la cuenta presente en la Base de Datos en la tabla Account con los datos de la cuenta recibida por parametros.

Los datos editables son: CSC, expirationDate, cardNumber y typeAccount.

1. Capa Business

Capa que separa la capa de datos de la capa de presentación. Para llevar a cabo esta función y la de validaciones se han creado las clases: AccountBusiness, DeleteAccount, insertAccount y updateAccount; las cuales de describen con mayor detalle a continuación.

1. AccountBusiness

Tiene como objetivo recibir las peticiones de la capa de presentación y realizar solicitudes a la capa Data relacionada a Cuenta, es decir a las funciones presentes en la clase AccountData.

También se le atribuye a esta capa la realización de validaciones.

Las funciones presentes en la clase AccountBusiness son:

* 1. getAllAccountAssetsBusiness(): Realiza la conexión con la función getAllAccountAssetsData() presente en la clase AccountData.
  2. insertAccountBusiness($account): Realiza la conexión con la función insertAccountData($account) presente en la clase AccountData.
  3. deleteAccountBusiness($idAccount): Realiza la conexión con la función deactivateAccountData($account) presente en la clase AccountData
  4. getIDBusiness():Realiza la conexión con la función getIDData() presente en la clase AccountData.
  5. getAccountByIdBusiness($idAccount): Realiza la conexión con la función getAccountByIdData($idAccount) presente en la clase AccountData.
  6. updateAccountBusiness($account): Realiza la conexión con la función updateAccountData($account) presente en la clase AccountData.
  7. validateEmpty($arrayVar): Valida que ninguno de los campos del array recibido por parametros este vacío.
  8. validateNumeric($arrayVar): Valida que todos los campos del array recibido por parametros sean de tipo numérico.

1. DeactivateAccount

Recibe el id de la cuenta que desea desactivar y haciendo el llamado de la función deactivateAccountBusiness($idAccount) presente en la clase AccountBusiness.

1. InsertAccount

Recibe por el id de la cuenta que se desea insertar, el id del ususario, a quien pertenece, CSC, tipo de cuenta, fecha de expiración número de tarjeta.

Para realizar el llamado de la función insertAccountBusiness($account) presente en la clase AccountBusiness.

1. UpdateAccount

Recibe el id de la cuenta que se desea actualizar, el id del usuario al cual le pertenece la cuenta, CSC, tipo de cuenta, fecha de expiración, numero de tarjeta. Para realizar el llamado de la función updateAccountBusiness($account) presente en la clase AccountBusiness.

1. Capa Presentation

Se creó la clase AccountInterface, la cal permite la interacción con el usuario y hace los llamados a las clases y funciones presentes en las clases de la capa business.  
Los datos correspondientes a cuenta mostrados en la presente plantilla son: CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

Se realizó el CRUD producto, en el cual se realizaron las siguientes clases y funciones: Michael Meléndez Mesén.

Capa Domain:

* Clase Product corresponde al objeto producto, el cual tiene los atributos correspondientes a: idProduct, brand, name, model, price, color, typeProduct, description y pathImages. Seguidamente cuenta con sus respectivos constructores, así como los set y get de cada atributo ya mencionados.

Capa Data:

* Clase Data: corresponde a la clase que se encarga de realizar la conexión con la base de datos, la misma cuenta con los siguientes atributos: server, user, password, db, dicha clase está compuesta por su correspondiente constructor.
* **Clase ProductData:** corresponde a la clase que cuenta con las funciones encargadas para el funcionamiento del CRUD producto, en la misma se hace una extensión de la clase Data y se incluyen las clases Data, Product y TypeProduct.

**Funciones:**

* + insertProduct($product): esta función es la que se encarga de realizar el registro del producto, recibe por parámetros un objeto de tipo Product; seguidamnete lo que se realiza es la obtención del ultimo idProduct registrado, para generar el siguiente idProduct, seguidamente por medio de una sentencia preparada se realiza la inserción en la base de datos, si el resultado es exitoso se retorna un true, de caso contrario false.
  + getProducts(): esta función se encarga de obtener todos los productos que se encuentran registrados en la base de datos, lo que se realiza es por medio de una sentencia preparada el select correspondiente a los productos con un ordenamiento ascendente, se retorna un arreglo de objetos de tipo Product.
  + updateProduct($product): esta función lo que permite es realizar la actualización de un producto en específico, recibe por parámetros un objeto de tipo Product, se crea la sentencia preparada para realizar la actualización correspondiente, los únicos parámetros que no podrán ser actualizados son el idProduct y el idTypeproduct. Si la actualización se realiza con éxito retorna true, de lo contrario false.
  + deleteProduct($id): esta función recibe por parámetros el idProduct que se desea eliminar, se realiza la sentencia preparada para dicha acción en la base de datos. Si la actualización se realiza con éxito retorna true, de lo contrario false.

Capa Business:

* Clase ProductBusiness: esta clase lo que corresponde es al llamado de las funciones que están realizadas en la clase ProductData, en la misma se incluyen las clases de ProductData y se encuentran las siguientes funciones:
  + insertProduct($product): realiza el llamado a la función y retorna lo que corresponde a insertProduct() correspondiente del DataProduct.
  + getProducts():realiza el llamado a la función y retorna lo que corresponde a getProducts() correspondiente del DataProduct.
  + updateProduct($product): realiza el llamado a la función y retorna lo que corresponde a updateProduct($product) correspondiente del DataProduct.
  + deleteProduct($id): realiza el llamado a la función y retorna lo que corresponde a deleteProduct($id) correspondiente del DataProduct.
* Clase ProductAction: en esta clase se realizan las acciones correspondientes al CRUD, dependiendo de la acción por realizar se ejecuta una función en específico.
  + Create Product: corresponde al registro del producto en la base de datos, se obtienen todos los valores correspondientes al formulario, se verifican que no existan inconsistencia en los tipos de datos y posterior se realiza la creación del objeto Product para seguidamente llamar la función insertProduct() correspondiente de la clase ProductBusiness y pasar por parámetros dicho objeto para su respectivo registro, si es exitoso se hace una redirección a la clase ProductCreate con un mensaje de éxito, de ser lo contrario se mostrará un mensaje de error.
  + Update Product: corresponde a la actualización del producto en la base de datos, se obtienen todos los valores correspondientes al formulario, se verifican que no existan inconsistencia en los tipos de datos y posterior se realiza la creación del objeto Product para seguidamente llamar la función updateProduct() correspondiente de la clase ProductBusiness y pasar por parámetros dicho objeto para su respectiva actualización, si es exitoso se hace una redirección a la clase ProductUpdate con un mensaje de éxito, de ser lo contrario se mostrará un mensaje de error.
  + Delete Product: corresponde la eliminación del producto en la base de datos, se obtienen el idproduct, posterior se realiza la llamada de la función deleteProduct() correspondiente de la clase ProductBusiness, se pasa por parámetros dicho idProduct para su respectiva eliminación, si es exitoso se hace una redirección a la clase ProductDelete con un mensaje de éxito, de ser lo contrario se mostrará un mensaje de error.

Capa Presentation:

* ProductCreate: corresponde a la plantilla HTML para realizar el registro de un producto, se muestra un formulario con los siguientes campos: marca, modelo, precio, color, tamaño y un botón para su respectivo registro, dicho formulario tiene como action la clase ProductAction.
* ProductUpdate: corresponde a la plantilla HTML para realizar la actualización de los productos que se encuentran registrados en la base de datos, se muestra un formulario con los siguientes campos: marca, modelo, precio, color, tamaño y un botón para su respectiva actualización, dicho formulario tiene como action la clase ProductAction. Dicha plantilla utiliza las funciones de getProducts() correspondiente a la clase ProductBusiness dicha debe estar incluida.
* ProductDelete: corresponde a la plantilla HTML para realizar la eliminación deun producto, se muestran los productos que están registrados en la base de datos, los valores que se muestran son: marca, modelo, precio, color, tamaño y un botón para su respectiva eliminación, dicho formulario tiene como action la clase ProductAction. Dicha plantilla utiliza las funciones de getProducts() correspondiente a la clase ProductBusiness dicha debe estar incluida.
* ProductRetrieve: corresponde a la plantilla HTML para visualizar todos los productos que se tienen registrados en la base de datos, se muestran los atributos de Brand, model, Price, color. Dicha plantilla utiliza las funciones de getProducts() correspondiente a la clase ProductBusiness dicha debe estar incluida.

Se realizó la documentación correspondiente a los casos de uso Producto. Encargado: Michael Meléndez Mesén.

* Se llevó a cabo la realización de script dinámicos de las direcciones de cliente que incluye las clases: DireccionClient,GetProvince,GetCanton,GetDistrict.

a) **Capa presentation:**

DirectionClient: Interfaz de usuario en la que se muestra el contenido de los

select de tipo provincia,canton y distrito.

b) **Capa data:**

Se creó la clase DirectionClient que contiene todos los métodos para el

cargado de los select de manera dinámica por medio de ajax, los nombres de los métodos son los siguientes: getProvincia(),getCanton() y getDistrito()

c) **Capa business:**

Archivos de php que son consumidos para el mostrado de la información de la en la capa de vista.

La tarea asignada de la creación de los scripts dinámicos de las direcciones fue realizada correctamente. Encargado: Anthony Hernández Badilla.

La tarea asignada correspondía a la realización del CRUD Celular y Computadora. Concluido, pero problema con carga del server GitHub. Albert Calderón Alvarado.

◦

• **Tareas a realizar.**

* CRUD de cuentas. Gustavo Najera
* CRUD de Producto. Michael Meléndez.
* CRUD de Computadora y Celular. Albert Calderón.
* Script de Clientes dirección. Anthony Hernández

• **Impedimentos**

* Problema de seguridad para registrar los cambios en el github online.

**2.** **Tareas realizadas que no fueron planeadas**.

◦ Se realizó el diseño y creación de la base de datos. Encargado: Michael Meléndez Mesén.

◦ Se realizó la creación y estructuración del proyecto en las diferentes capas Business, Data, Domain y Presentation. Encargado: Michael Meléndez Mesén

**3.** **Asuntos Pendientes**

* Tarea 3 CRUD de Computadora y Celular. que aún no se ha resuelto y siguen postergándose

**4.** **Asuntos Nuevos**

* Se planeó que la tienda virtual ya no sería especializada en dos productos, sino algo más general, es por eso que la creación del CRUD computadora y celular fueron eliminados, así como las tablas en la base de datos.
* Se procede a crear el CRUD de tipo producto y la creación de la tabla en la base de datos. Tarea asignada a Albert Calderón Alvarado.

**5.** **Cumplimiento de tareas asignadas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Responsable** | **Grado de conclusión** | **Nota** |
| CRUD Celular y Computadora | Albert Calderón Alvarado | Incompleto |  |
| CRUD Cuenta | Gustavo Nájera Nájera | Completo. |  |
| CRUD de clientes dirección | Anthony Hernández Badilla | Completo |  |
| CRUD Producto | Michael Meléndez Mesén | Completo |  |

\*Aunque el proyecto es grupal, en cada iteración se asignará una persona del grupo como responsable de cada una de las tareas asignadas.

\*\*Existen tareas ya definidas como avances revisables en cada iteración:

1. Diagramas de casos de uso

2. Modelo de datos

3. Diagrama de aplicación

4. Módulos funcionales en ejecución

**PRÓXIMA REUNIÓN:**

Fecha: 16 de enero

Hora: 8:00 AM

Lugar: Módulo Indígena Sede del Atlántico UCR -Turrialba

**CLAUSURA:**

Alberth Calderon Alvarado, como persona encargada de preceder la reunión, clausura la reunión a las 10: 20 horas.

**MINUTA**

Minuta No. 002 MGASoluciones

Fecha : Lunes 16 de enero

Hora : 09:50 am

Lugar : Laboratorio C, UCR Sede del Atlántico

Siendo las 09:50 horas del día 16 de enero de 2016, se reúne el grupo:**MGASoluciones**, en sesión extraordinaria previa citación realizada.

**ASISTENTES:**

La persona encargada que precede la reunión, Michael Meléndez Mesén, pasó lista y determinó que había quórum.

Los asistentes a esta reunión son:

· Albert Calderón Alvarado. \_\_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_

· Anthony Hernández Badilla \_\_\_\_\_ Presente\_\_\_\_\_\_\_\_\_

· Gustavo Nájera Nájera. \_\_ \_Presente\_\_\_\_\_\_\_\_\_

· Michael Meléndez Mesén. \_\_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_

**AGENDA:**

1. Entrega de tareas a los miembros del grupo.

**ASUNTOS:**

**1.** **Asuntos tratados y acuerdos tomados**

* Agregar atributo nombre al CRUD producto **Michael Meléndez Mesén**
* Agregar imágenes al producto (6 máximo). **Michael Meléndez Mesén**
* Criterios para calificar productos **Albert Calderón Alvarado**
* CRUD búsquedas **Gustavo Nájera Nájera**.
* CRUD cliente **Albert Calderón Alvarado**
* Funcionalidad del muro de cada producto en específico **Anthony Hernández Badilla**
* Máscara para campo precio en producto. **Michael Meléndez Mesén**
* Sesiones para obtener idClient **Michael Meléndez Mesén**

• **Tareas concluidas con éxito.**

1. Se realizó la actualización del CRUD producto, (se agregaron campos correspondientes al tipo producto, nombre e inserción de imágenes). Michael Meléndez Mesén.

Capa Data:

* ProductData: funciones actualizadas y nuevas:
  + insertProduct($product,$arrayImages): en esta actualización dicha función recibe por parámetros un arreglo con la ruta de las imágenes que corresponden a un producto en específico, se realiza el registro de la información en la tabla tbProduct, seguidamente el registro de las imágenes en la tabla tbImageProduct. Si los registros se realizan con éxito se retorna true, de ser lo contrario false.
  + insertImageProduct($idProduct, $arrayPath): función que se encarga de registrar nuevas imágenes a un producto en específico, recibe por parámetros el idProduct y un arreglo con las rutas de las imágenes, el mismo no puede exceder el máximo de 6 ya que ese es el límite por artículo, se realiza la sentencia preparada para realizar el respectivo registro, así como la generación del idImageProduct consecutivo al último registro que exista. Si el registro se realiza con éxito se retorna true de lo contrario false.
  + deleteImageProduct($idProduct, $path): función que permite eliminar imágenes que corresponden a un producto, recibe por parámetros el idProduct y la path, con los cuales se realiza la sentencia preparada para realizar la eliminación de dicha imagen. Si la eliminación se realiza con éxito retorna true, de la contrario false.

Capa Business:

* ProductBusiness: funciones actualizadas y nuevas
  + insertProduct($product,$arrayImages): realiza el llamado a la función y retorna lo que corresponde a insertProduct() correspondiente del ProductData.
  + insertImageProduct($idProduct, $arrayPath): realiza el llamado a la función y retorna lo que corresponde a insertImageProduct() del ProductData.
  + deleteImageProduct($idProduct,$path): realiza el llamado a la función y retorna lo que corresponde a deleteImageProduct del ProductData.

Capa Presentation:

* ProductCreate: se le agregó al formulario los campos de nameproduct y un input tipo file para agregar imágenes, también se creó un script tipo java script llamado GenerateFields en la carpeta JS; que permite agregar los input para las imágenes dinámicamente, por lo cual se podrán agregar hasta el límite de 6 imágenes al momento de registrar el producto. Por otra parte se agregó una máscara moneda para el campo de precio.
* ProductUpdate: se agregó el campo nameproduct para poder ser actualizado, además se pueden actualizar las imágenes correspondientes al producto, (eliminar y registrar), si el producto no cuanta con el límite de 6 imágenes se le da la opción de agregar más. Por otra parte se agregó una máscara moneda para el campo de precio.
* ProductInsertImage: corresponde a la plantilla HTML para ingresar imágenes a un producto, se utiliza un script java script llamado GenerateFieldsImages que se encuentra en la carpeta JS, el cual lo que recibe es la cantidad de imágenes que tiene registrado un producto para sí permitir hasta el máximo de 6, el script genera los input tipo file de manera dinámica hasta llegar al límite.
* ProductDelete: se agregó el campo nameProduct para visualizar mejor la información del producto que se desea eliminar. Por otra parte se agregó una máscara moneda para el campo de precio.
* ProductRetrieve: Se agregó una máscara moneda para el campo de precio.

Se realizó el manejo de sesiones para obtener el idCliente, para posteriormente ser registrado en la tabla de búsquedas. Encargado: Michael Meléndez Mesén

Capa Data:

* ClientLoginData: corresponde a la clase que realiza la función para verificar si el cliente que intenta ingresar existe.
  + isClient($client): función que realiza la consulta en la base de datos para verificar si el cliente existe, recibe por parámetros los valores de userName y password, se realiza la sentencia preparada y se consulta en la base de datos, si la consulta fue exitosa retorna el id del cliente, de lo contrario -1.

Capa Business:

* ClientLoginBusiness:
  + isClient($client): realiza el llamado a la función y retorna lo que corresponde a isClient() correspondiente del ClientLoginData.
* LoginAction: clase que obtiene los valores que son enviados del formulario correspondiente al login, se hace uso de la función isClient de la clase ClientLoginBusiness, si el retorno es exitoso se crea una variable de sesión con el idClient, y es re direccionado al index, de lo contrario se devuelve al login con un mensaje de error.

Capa Presentation:

* Index: se agregó un formulario con los capos de userName y password para realizar el login correspondiente, dicho formulario tiene como action la clase LoginAction que realiza la función específica según sea el caso.

Se realizó la actualización de la documentación correspondiente a los casos de uso Producto. Encargado: Michael Meléndez Mesén.

1. Se realizó la incorporación de búsquedas de productos. Encargado: Gustavo Nájera Nájera

Capa Data

Clase SearchData: Realiza las conexiones en la Base de datos que corresponden a las búsquedas. Las funciones que contiene son:

searchProductData($termSearch): Recibe un término de búsqueda el cual será comparado con los datos de las cuentas activas e insertadas previamente en la Base de Datos en los campos descripción, modelo, marca y nombre del producto.

insertSearchData($arraySearch): Realiza la inserción en la tabla tbSearch, de todos los productos que tuvieron coincidencia en la búsqueda.

Los productos son recibidos por parametros en un array y son relacionados al cliente que ha iniciado sesión.

Capa Business

Clase SearchProductBusiness

Función searchProduc($termSearch): Realiza la conexión con el método searchProductData($termSearch) presente en la clase SearchData para realizar la búsqueda pertinente.

Capa Presentation

Se crea la plantilla html para la interacción con el usuario en el cual se da la posibilidad de ingresar un término de búsqueda, en caso de encontrar coincidencias con los productos ingresados previamente en la base de datos, los datos de cada producto a mostrar son: id del producto, precio, nombre, modelo, precio, color, descripción, tipo de producto, y todas las imágenes de dicho producto.

1. Se actualizaron los atributos referentes a Cuenta. Encargado: **Gustavo Nájera Nájera**

Capa Domain

Se actualizan los atributos de cuenta, quedando solamente con los siguientes datos: CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

Capa Data

Clase Account Data.

Función getAllAccountAssetsData(): La creación de los objetos para cada cuenta obtenida de la Base de Datos es modificado, para utilizar solamente los atributos CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

Función getAccountByIdData($idAccount): La creación del objeto creado en caso de encontrar coincidencias es modificado para utilizar solamente los atributos CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

Función insertAccountData($account): La consulta para realizar la insersion de la nueva cuenta en la Base de Datos es modificada para tomar en cuenta solamente los atributos actualizados es decir: CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

Función deleteAccountData ($idAccount): Realiza la eliminación de la cuenta insertada previamente en la Base de Datos que coincida con el id de la cuenta recibido por parametros.

Función updateAccountData($account): Realiza la actualización de los datos CSC, expirationDate, cardNumber y typeAccount en la cuenta ingresada previamente en la Base de Datos que coincida con el id de la cuenta recibida por paramentros.

Capa Business

Clase InsertAccount

Se modifica para que solo reciba los siguientes datos: CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

Clase UpdateAccount

Se modifica para que solo reciba los siguientes datos: CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.

Capa presentación

Clase AccountInterface

Se modifica para que solo muestre los siguientes datos Se modifica para que solo reciba los siguientes datos: CSC, expirationDate, idClient, idAccount, cardNumber y typeAccount.  
También en la sección de actualización, es modificado para que solo permita la edición de los atributos CSC, expirationDate, cardNumber y typeAccount.

1. Se reestructura el proyecto en los archivos relacionados a Cuenta y Búsquedas. Encargado: Gustavo Nájera Nájera

Capa Business

Se crea la carpeta, Account, en la cual son ingresadas las clases: AccountBusiness, DeactivateAccount, InsertAccount, UpdateAccount. Por lo que las redirecciones son modificadas en cada una de estas clases, asi como en la clase.

Capa presentación.

Se crea la carpeta Account, en la cual se ingresa la clase AccountInterface, por lo que se modifican las redirecciones en la misma.

◦ Se crea el CRUD de Tipo de Producto. Encargado Albert Calderón Alvarado

◦ Se crea el CRUD de Cliente. Encargado Albert Calderón Alvarado.

◦ Se eliminó el CRUD Celular y Computadora (Innecesarias al normalizar la tabla producto). Encargado Albert Calderón Alvarado

◦ Se creó el muro general donde se pueden registrar y ver los comentarios que se han hecho a los productos de la página. Anthony Hernández Badilla

◦ Se creó el muro para computadoras donde se pueden registrar y ver los comentarios que se han hecho referentes al artículo computadora. Anthony Hernández Badilla

◦ Se creó el muro para computadoras donde se pueden registrar y ver los comentarios que se han hecho referentes al artículo mencionado. Anthony Hernández Badilla

◦ Se creó el muro para celulares donde se pueden registrar y ver los comentarios que se han hecho referentes al artículo mencionado. Anthony Hernández Badilla

◦ A pesar de que se eliminó la tabla computadora y celular se crearon los muros correspondientes para estos, sin embargo se debe migrar a muros dinámicos para los diferentes productos con los que cuenta la tienda.

• **Tareas a realizar.**

* Actualización CRUD de Producto. Michael Meléndez.
* Creación de máscara para el campo precio. Michael Meléndez Mesén
* Manejo de sesiones para obtener el idCliente. Michael Meléndez Mesén
* Incorporar la funcionalidad de búsquedas. Gustavo Nájera Nájera
* Actualizar los atributos referentes a Cuenta. Gustavo Nájera Nájera.
* Agregar el CRUD Cliente a la funcionalidad. Albert Calderón Alvarado.
* Sumar el CRUD Tipo de Producto al proyecto. Albert Calderón Alvarado.
* Limpiar del proyecto todo lo referente y exclusivo de Computadora y Celular. Albert Calderón Alvarado.
* Creación de muros dinámicos para el mostrado e insertado de información referente al producto seleccionado.. Anthony Hernández Badilla

• **Impedimentos**

◦ Debido a que se migró el proyecto a que la tienda posee diferentes productos, no se logró cambiar a tiempo la creación de muros dinámicos para el insertado y mostrado de información, tarea a retomar en el próximo entregable.

**2.** **Tareas realizadas que no fueron planeadas**.

◦ Se realizó la actualización de la base de datos (Homologar todas las tablas, así como la creación de nuevas con las diversas relaciones, se cambió todo la base al idioma inglés). Encargado: Michael Meléndez Mesén.

◦ Se reestructura el proyecto en los archivos relacionados a Cuenta. Encargado: Gustavo Nájera Nájera

**3.** **Asuntos Pendientes**

* CRUD de Cliente se encuentra incompleto ya que la dirección no se está generando utilizando las tablas de provincia, cantón y distrito. **Albert Calderón Alvarado.**
* Criterios de calificación incompletos. No se realizó **Albert Calderón Alvarado.**
* CRUD Muro se encuentran incompletos ya que no están creados dinámicamente, solo el muro general se encuentra funcionando**. Anthony Hernández Badilla**

**4.** **Asuntos Nuevos**

**5.** **Cumplimiento de tareas asignadas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Responsable** | **Grado de conclusión** | **Nota** |
| Actualización CRUD Producto.  Creación de máscara, campo precio.  Manejo de sesiones para obtener idCliente | Michael Meléndez Mesén | Completo |  |
| Actualización de CRUD Cuenta.  Incorporación de Búsquedas. | Gustavo Nájera Nájera | Completado |  |
| Creación del CRUD Cliente y CRUD Tipos de Producto. | Albert Calderón Alvarado | Incompleto |  |
| Criterios de calificación por producto | Albert Calderón Alvarado | Sin realizar |  |
| Creación de muro general y de productos de la tienda | Anthony Hernández Badilla | Incompleto |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*Aunque el proyecto es grupal, en cada iteración se asignará una persona del grupo como responsable de cada una de las tareas asignadas.

\*\*Existen tareas ya definidas como avances revisables en cada iteración:

1. Diagramas de casos de uso

2. Modelo de datos

3. Diagrama de aplicación

4. Módulos funcionales en ejecución

**PRÓXIMA REUNIÓN:**

Fecha: 23 de enero

Hora: 8:00 AM

Lugar: Laboratorio C, Sede del Atlántico UCR -Turrialba

**CLAUSURA:**

Michael Meléndez Mesén, como persona encargada de preceder la reunión, clausura la reunión a las 10: 20 horas.

**MINUTA**

Minuta No. 003 MGASoluciones

Fecha: Miércoles 18 de enero

Hora: 2:00 pm

Lugar: Laboratorio C, UCR Sede del Atlántico

Al ser las 2:00pm del día miércoles 18 de enero de 2017, se reúne el grupo: **MGASoluciones**, en sesión extraordinaria previa citación realizada.

**ASISTENTES:**

La persona encargada que precede la reunión, Gustavo Nájera Nájera, pasó lista y determinó que había quórum.

Los asistentes a esta reunión son:

· Albert Calderón Alvarado. \_\_\_Presente\_\_\_

· Anthony Hernández Badilla \_\_ Presente\_\_\_

· Gustavo Nájera Nájera. \_\_Presente\_\_\_\_

· Michael Meléndez Mesén. \_\_ Presente\_\_\_\_

**AGENDA:**

Entrega de tareas a los miembros del grupo.

**ASUNTOS:**

1. **Asuntos tratados y acuerdos tomados**
2. **Tareas concluidas con éxito.**

* Se incluye el script de predicción en las búsquedas. **Gustavo Nájera Nájera**

**Capa Business**

Clase GetAllBusinessForAJAX: consulta al método getProducts(), que pertenece a la clase productBusiness.  
se utilizan los métodos deleteSpecialCharacters($text) y normalVowels($cadena) para eliminar los caracteres especiales y las tildes respectivamente.

**Capa Presentation**

Clase Search: Se inserta el script de predicción tomando en cuenta los datos obtenidos por Ajax desde el la clase GetAllBusinessForAJAX, la cual pertenece a la capa Business.

* Se crea la plantilla HTML para detalles de productos. **Gustavo Nájera Nájera**

**Capa Data**

En la clase **Clase ProductData** se crea la función getProductByID, la cual realiza la consulta en la Base de datos para obtener el producto con el id recibido por parámetros.

**Capa Business**

En la **Clase ProductBusiness** se crea la función getProductByID, la cual consulta al método getProductByID que pertenece a la clase ProductData en la capa Data.

**Capa Presentation**

Se crea la plantilla HTML **ProductDetails**, con el fin de mostrar los detalles de un producto específico.

Se obtienen los detalles de un producto realizando la consulta al método getProductByID que pertenece a la clase ProductBusiness en la capa Business.

**Clase Search**: Se enlazan los resultados de las búsquedas con la opción de ver detalles, es decir redirecciona a la página de ProductDetails.

* Búsquedas solo puede tomar en cuenta productos activos. **Encargado Gustavo Nájera**

**Capa data**

En la función **Clase SearchData** se modifica la función searchProductData, la consulta a la Base de Datos con el fin de tomar en cuenta el atributo active, obteniendo solo los productos activos.

* Agregar atributo activo en Cuenta. **Encargado Gustavo Nájera**

Se modifica la base de datos agregando el atributo active.

* Correcciones en los Casos de Uso. **Encargado Gustavo Nájera**

Se modifican los casos de uso referentes a Cuenta y a búsqueda, con el fin de incluir los cambios realizados en el código.

* El sistema debe funcionar con la BD en inglés. **Encargado Gustavo Nájera**

**Capa Data.**

**Clase AccountData**

En funciones getIDData(), getAllAccountAssetsData(), getAccountByIdData($idAccount), insertAccountData($account), deactivateAccountData($idAccount) y updateAccountData($account) se modifican las consultas en la Base de datos para que utilicen las Base de Datos en inglés.

* Crear las tablas VentasConcretadas, Ventas canceladas y Lista de deseos. **Encargado Gustavo Nájera**

Se crean las tablas VentasConcretadas, Ventas canceladas y Lista de deseos en la Base de Datos en inglés.

* En los resultados de las búsquedas debe mostrar menos datos. **Encargado Gustavo Nájera**

**Capa Data**

**Clase SearchData**

Función searchProductData se modifica con el fin de que solo obtenga una imagen de cada producto.

**Capa Presentación**

Clase Search

Se modifica con el fin de que solo muestre los datos: Nombre, precio, marca y modelo.

Se realizó el CRUD compra cliente, en el cual se realizaron las siguientes clases y funciones: Michael Meléndez Mesén.

Capa Domain:

Ø Clase CustomerShopping: corresponde al objeto compra cliente, el cual tiene los atributos correspondientes a: idClient, datePurchase, totalPurchase, idSale. Seguidamente cuenta con sus respectivos constructores, así como los set y get de cada atributo ya mencionados.

Capa Data:

Ø **Clase CustomerShoppingData:** corresponde a la clase que cuenta con las funciones encargadas para el funcionamiento del CRUD compra cliente, en la misma se hace una extensión de la clase Data y se incluyen las clases Data, CustomerShopping.

**Funciones:**

o insertCustomerInvoice($customerShopping,$products): esta función es la que se encarga de realizar el registro de la compra cliente, recibe por parámetros un objeto de tipo CustomerShopping y un arreglo con los idProduct; seguidamente lo que se realiza es la obtención del ultimo idSale registrado, para generar el siguiente idSale, posterior por medio de una sentencia preparada se realiza la inserción en la base de datos, si el resultado es exitoso se retorna un true, de caso contrario false.

o cancelInvoice($id): esta función se encarga de realizar la anulación de una compra cliente, lo que recibe por parámetros es el id que corresponde a la compra cliente, se realiza la sentencia reparada con la actualización del atributo active con valor 1 para indicar que esa compra cliente fue anulada, si el resultado es exitoso se retorna un true, de caso contrario false.

o getCustomerInvoice($idSale): esta función lo que permite es obtener los datos de una compra cliente en específico, recibe por parámetros el idSale, con el cual se realiza la sentencia preparada, para consultar en la base de datos, si la consulta es exitosa se retorna un objeto de tipo CustomerShopping, así como uno de cliente, de lo contrario se retorna false.

o getCustomerInvoices(): esta función se encarga de obtener todas las compras cliente que se hayan registrado, si la consulta se realiza con éxito retorna un arreglo con objetos de tipo CustomerShopping y Client.

Capa Business:

Ø Clase CustomerShoppingBusiness: esta clase lo que corresponde es al llamado de las funciones que están realizadas en la clase CustomerShoppingData, en la misma se incluyen las clases de CustomerShoppingData y se encuentran las siguientes funciones:

o insertCustomerInvoice($customerShopping,$products): realiza el llamado a la función y retorna lo que corresponde a insertCustomerInvoice($customerShopping,$products) correspondiente del CustomerShoppingData.

o cancelInvoice($id): realiza el llamado a la función y retorna lo que corresponde a cancelInvoice($id) correspondiente del CustomerShoppingData.

o getCustomerInvoice($idSale): realiza el llamado a la función y retorna lo que corresponde a getCustomerInvoice($idSale) correspondiente del CustomerShoppingData.

o getCustomerInvoices(): realiza el llamado a la función y retorna lo que corresponde a getCustomerInvoices(): correspondiente del CustomerShoppingData.

Ø Clase CustomerShoppingAction: en esta clase se realizan las acciones correspondientes al CRUD, dependiendo de la acción por realizar se ejecuta una función en específico.

o Create CustomerShopping: corresponde al registro del la compra del cliente en la base de datos, se obtienen todos los valores correspondientes al formulario, se verifican que no existan inconsistencia en los tipos de datos y posterior se realiza la creación del objeto CustomerShopping, así como un arreglo con los idProduct, para seguidamente llamar la función insertCustomerInvoice($customerShopping,$products) correspondiente de la clase CustomerShoppingBusiness y pasar por parámetros dicho objeto para su respectivo registro.

o Delete Product: corresponde la eliminación del producto en la base de datos, se obtienen el idproduct, posterior se realiza la llamada de la función deleteProduct() correspondiente de la clase ProductBusiness, se pasa por parámetros dicho idProduct para su respectiva eliminación, si es exitoso se hace una redirección a la clase ProductDelete con un mensaje de éxito, de ser lo contrario se mostrará un mensaje de error.

Capa Presentation:

Ø TestCustomerShopping: plantilla HTML para realizar la prueba de las diversas funciones del CRUD compra cliente, se realizó dicha plantilla ya que la parte correspondiente al carrito de compras asignada a **Albert Calderón Alvarado** no se ha realizado.

Ø CustomerShoppingState: plantilla HTML para realizar la anulación de las compras cliente, se muestra el número de compra, nombre del cliente, fecha compra y total de compra, con su respectivo botón para realizar la anulación de dicha compra.

Encargado Anthony Hernández Badilla

**Tareas completadas con éxito:**

Se retomó la creación de muros dinámicos para cada producto con toda la funcionalidad requerida, además de ordenar por capas las funcionalidades anteriores junto con las desarrolladas.

Para la actual iteración se realizó la funcionalidad de incluir la tabla frecuencia para registrar los inicios de sesión, la fecha en la que inició sesión, las intervenciones en muros, las búsquedas de productos y visitas de productos a estos.

Distribución de clases de la funcionalidad de muros dinámicos:

**Capa Presentation:**

Contiene la clase ProductOption en la que se cargan los select de la clasificación del producto y el tipo de producto para que el usuario pueda seleccionar el muro en el que quiere insertar la información.

Una vez seleccionado el muro en el que quiere insertar un comentario se le cargará la clase Wall en la que podrá visualizar todos los comentarios insertados para ese producto e insertar uno nuevo.

**Capa domain:**

Contiene la clase Comment que contiene los atributos solicitados que van a ser ingresados en la tabla tbcomment.

**Clase data**:

Contiene la clase ProductWall, que va contener las funciones getTipoProducto y getProducto, que se encargan de enviar las peticiones por ajax para el cargado de los select.

Contiene las clase WallBusiness que contiene los métodos getAllComments e insertCommet, el primero se encarga de extraer todos los comentarios de la base de datos referentes al muro que se seleccionó previamente y el segundo método se encarga de insertar un registro en la base de datos de un comentario con un nuevo id de comentario,el producto al que pertenece, el comentario, el id de cliente y la fecha en la que se realizó.

**Capa Business:**

La clase WallBusiness extiende de la clase data WallBusiness que contiene todos los métodos para la búsqueda e insertado de comentarios.

Además posee un archivo php insertComment que se encarga de hacer las

validaciones pertinentes para el insertado de comentarios y de redireccionar a la vista de selección de comentarios.

Además posee los archivos php GetProduct.php y GetTypeProduct.php que se encargan de consumir la información recibida de la clase ProductWall para el cargado de los select de manera dinámica.

Funcionalidad de frecuencia de usuario.

**Capa data:**

Contiene la clase Frecuency que contiene el método createFrecuency que se encarga de insertar un nuevo registro en la base de datos de una sesión iniciada con los atributos de un identificador,fecha, id de cliente, y tres valores inicializados en cero que corresponden a la cantidad de veces que ha comentando un muro,las búsquedas realizadas y las veces que ha visitado un producto, todo esto dentro de la tabla tbfrecuency.

UpdateWall: Método que se encarga de obtener el número de identificador del usuario que ha iniciado sesión,esto para poder poder realizar la actualización del campo participationWall cuando el usuario ha ingresado un nuevo comentario.

UpdateSearch: Método que se encarga de obtener el número de identificador del usuario que ha iniciado sesión,esto para poder poder realizar la actualización del campo searchProduct cuando el usuario ha realizado una nueva búsqueda

ViewWall:Método que se encarga de obtener el número de identificador del usuario

que ha iniciado sesión,esto para poder poder realizar la actualización del campo

viewProduct cuando el usuario ha visitado un producto.

**Capa Business:**

FrecuencyBusiness: Clase que extiende de la clase Frecuency para poder utilizar los métodos creados en la capa de data.

Encargado Alberth Calderón Alvarado.

Tareas con Tipo Producto, Cliente, ProductoDeseado.

**SQL:** Se agregó el atributo active a las tablas tbtypeproduct, client y tbproductdesired ,y el atributo dateactive a la tabla tbproductdesired.

**Capa Data:**

**Tipo Producto Data:**

* insertTypeProduct($typeProduct): Se modifica la función para que éste aplique el dato active y trabaje con la tabla en inglés.
* getTypeProduct(); Se modifica para que trabaje con la base en inglés y que obtenga un array de los tipos de productos que están registrados y con el atributo active=1
* deleteTypeProduct($idTypeProduct); Se modifica la función para que esta no elimine tuplas de la base de datos, si no que a su vez actualice la tupla con el atributo active=0
* updateTypeProduct($typeProduct); Función actualiza los datos de las tuplas, trabaja con la base en inglés.

**Cliente Data:**

* insertClient($client): Se modifica la función para que aplique el dato active y trabaje con la tabla en inglés.
* getClient(); Se modifica para que trabaje con la base en inglés y que obtenga un array de los clientes que están registrados y con el atributo active=1
* updateClient($client); Función recibe un archivo cliente de la clase domain, este posee un id registrado, y actualiza la tupla del gestor de base de datos con los recibidos en esta función.
* deleteClient($idClient); Se modifica la función para que esta no elimine tuplas de la base de datos, si no que a su vez actualice la tupla con el atributo active=0

Detalles de Producto.

**Detalle Producto Data:**

* insertDesire($idProductWish, $idclientWish): La función consulta el id de mayor tamaño, además consulta si existe ya registrado la tupla con el id de producto y el id cliente en sesión, si existe actualiza su estado a activo y modifica fecha, si no existe inserta una tupla nueva.
* isDesired($idProductWish, $idclientWish); consulta si existe ya registrado la tupla con el id de producto y el id cliente en sesión, y si está active=1. Devuelve un valor booleano.
* deleteDesire($idProductWish, $idclientWish) Se modifica la función para que esta no elimine tuplas de la base de datos, si no que a su vez actualice la tupla con el atributo active=0 y actualiza la fecha de modificación.

**Capa** **Business**

**Detalle Producto**

**details Business**  hace override de los métodos de Data y un constructor.

**Acción deseado** (desiredAction); se encarga de controlar si el producto esta deseado y la acción del botón para cambiar la opción de deseo.

**Capa Presentación**

**Detalle de producto:** Se agrega al Html productDetails, un checkbox que muestra si el producto ha sido deseado o no por el cliente en session. También se agrega un botón con la opción de cambiar el deseo entre activo y desactivo varía del estado en el que se encuentra.

1. **Tareas a realizar.**
   * Tareas a realizar por **Gustavo Nájera Nájera.**
     + - Incluir el script de predicción en las búsquedas.
       - Crear plantilla HTML en la cual se mostrarán todos los detalles de un producto específico.
       - Mostrar solo resultados de productos activos en la búsqueda de productos.
       - Agregar atributo activo en la tabla cuenta en la Base de Datos.
       - Cambiar el código referente a búsqueda, sesión, y cuenta para que funcione con la Base de Datos en Ingles.
       - Realizar las correcciones pertinentes en los Casos de Uso.
       - Crear las tablas: VentasConcretadas, Ventas canceladas y Lista de deseos.
       - En los resultados de búsqueda se mostraran solo los datos nombre, marca, modelo y precio.

* Tareas a realizar por **Alberth Calderon Alvarado.**
  + - * Agregar el atributo active a Tipo producto, modificación de su CRUD y actualización con las tablas en ingles.
      * Agregar el atributo active a Cliente, modificación de sus capas y funciones y actualización con las tablas en ingles.
      * Implementación de los productos deseados.
      * Agregar el atributo activo y Fecha de Modificación a Producto deseado.

• **Impedimentos**

1. **Tareas realizadas que no fueron planeadas**.

En la plantilla referente a cuenta, se cambia el input de cuenta de tipo texto a tipo date y se hace no editable el campo de id de cliente. **Encargado: Gustavo Nájera Nájera**

1. **Asuntos Pendientes**
2. **Asuntos Nuevos**
3. **Cumplimiento de tareas asignadas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Responsable** | **Grado de conclusión** | **Nota** |
| Se creó el CRUD compra cliente, se modificó opción de eliminar producto, se cambió a desactivar producto, se agregó campo state en la tabla tbProduct | Michael Meléndez Mesén | Concluido |  |
| Incluir script de predicción en las búsquedas. | Gustavo Nájera Nájera | Concluido |  |
| Crear plantilla HTML para detalles de productos. | Gustavo Nájera Nájera | Concluido |  |
| Búsquedas solo puede tomar en cuenta productos activos. | Gustavo Nájera Nájera | Concluido |  |
| Agregar atributo activo en Cuenta. | Gustavo Nájera Nájera | Concluido |  |
| El sistema debe funcionar con la BD en inglés. | Gustavo Nájera Nájera | Concluido |  |
| Correcciones en los Casos de Uso. | Gustavo Nájera Nájera | Concluido |  |
| Crear las tablas: VentasConcretadas, Ventas canceladas y Lista de deseos. | Gustavo Nájera Nájera | Concluido |  |
| En los resultados de las búsquedas debe mostrar menos datos. | Gustavo Nájera Nájera | Concluido |  |
| Modificar la tabla y capas de tipo producto | Albert Calderón Alvarado | Concluido |  |
| Agregar “activo” a cliente(modificación del CRUD) | Albert Calderon Alvarado | Concluido |  |
| Agregar la opcion de producto deseado al detalle de producto | Alberth Calderon Alvarado | Concluido |  |
| Implementación del producto deseado, en capas,base de datos, y preparación para mineria | Alberth Calderon Alvarado | Concluido |  |
| Creación de muros para cada tipo de producto. | Anthony Hernández Badilla | Concluido |  |
| Sincronización de select para poder seleccionar el muro deseado. | Anthony Hernández Badilla | Concluido |  |
| Creación de la tabla comment | Anthony Hernández Badilla | Concluido |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Creación de la tabla frecuency | Anthony Hernández Badilla | Concluido |  |
| Creación de nuevo registro en la tabla frecuency al iniciar sesión desde la página. | Anthony Hernández Badilla | Concluido |  |
| Actualización de las participaciones en muros de un usuario | Anthony Hernández Badilla | Concluido |  |
| Actualización de las búsquedas realizadas por un usuario | Anthony Hernández Badilla | Concluido |  |
| Actualización de las vistas de un producto | Anthony Hernández Badilla | Concluido |  |

\*Aunque el proyecto es grupal, en cada iteración se asignará una persona del grupo como responsable de cada una de las tareas asignadas.

\*\*Existen tareas ya definidas como avances revisables en cada iteración:

1. Diagramas de casos de uso

2. Modelo de datos

3. Diagrama de aplicación

4. Módulos funcionales en ejecución

**PRÓXIMA REUNIÓN:**

Fecha: 30 de enero

Hora: 8:00 AM

Lugar: Laboratorio C, Sede del Atlántico UCR -Turrialba

**CLAUSURA:**

Gustavo Nájera Nájera, como persona encargada de preceder la reunión, clausura la reunión a las 3:00pm.

**MINUTA**

Minuta No. 004 de MgaSoluciones

Fecha : 30 enero de 2107

Hora : 8:30

Lugar : Laboratorio c, UCR, Sede del atlántico

Siendo las 2pm horas del día 30 de enero del 2017, se reúne el grupo: **MGASoluciones**, en sesión extraordinaria previa citación realizada.

**ASISTENTES:**

La persona encargada que preceder la reunión, Anthony Hernández, pasó lista y determinó que había quórum.

Los asistentes a esta reunión son:

· Albert Calderón Alvarado \_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_\_\_\_

· Anthony Hernández Badilla \_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_\_\_\_

· Gustavo Nájera Nájera \_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_\_\_\_

· Michael Meléndez \_\_\_\_\_\_Presente\_\_\_\_\_\_\_\_\_\_\_\_\_

**AGENDA:**

**ASUNTOS:**

**1.** **Asuntos tratados y acuerdos tomados**

• **Qué he hecho**

* Mejorar script de predicción en búsquedas. **Encargado Gustavo Nájera Nájera**
* Implementar el sistema de likes en muros. Encargado Anthony Hernandez Badilla

**Capa Data**

En la clase **SearchData**, se modificó la función:

searchProductAutompleteData, con el fin de que el autocompletado funcione de manera independiente por tipo de dato, es decir, cada palabra escrita busca coincidencias en tablas diferentes.

En la clase **SearchData**, se creó la función:

getQuery, con el objetivo de que genere la consulta a ejecutar en la Base de Datos relacionada a búsquedas.

En la clase **SearchData**, se modificó la función:

searchProductData con el objetivo de que utilice la consulta obtenida de la función getQuery.

**Capa Business**

Se modifica la clase **GetAllBusinessForAjax** destinada a realizar consultas

por ajax para la función de autocompletado en búsquedas, con el fin de que sólo devuelva resultados del atributo correcto.

**Capa Presentación**

Se modifica la clase Search, específicamente el script javascript encargado de realizar la consulta por ajax para el autocompletado con el fin de que solicite resultados relacionados solamente a un atributo en específico de los productos.

* Almacenar la dirección al registrar la cuenta. **Encargado Gustavo Nájera Nájera**.

**Capa Data**

En la clase **AccountData** se toma en cuenta el atributo dirección para realizar las consultas a la Base de Datos.

**Capa Domain**

En la clase Account se incluye el atributo dirección.

**Capa Business.**

En la clase **InsertAccount** se incorpora el atributo dirección, sin embargo, el valor se coloca por defecto, ya que se debe obtener del cliente y dicho valor no ha sido incorporado aún a la información almacenada de los cliente.

En la clase **updateAccount** se modifica la creación del objeto cuenta para que tome en cuenta el atributo dirección.

* Pasar búsquedas a la vista cliente. **Encargado Gustavo Nájera**

**Capa Data**

En la clase **SearchData** se modifica la dirección de impartición de clases.

**Capa Presentacion**

En la clase **ClientView** se agrega el script que ejecuta la función de autocompletado, importaciones y código que muestra resultados de las búsquedas.

* Validaciones en cuenta del lado cliente. **Encargado Gustavo Nájera**

**Capa Presentación**

En la clase AccountInterface se agrega código javascript encargado de validar y se modifica el formulario para dicha funcionalidad.

* Crear y utilizar tabla en BD tipoCuenta. **Encargado Gustavo Nájera**

Se crea la tabla tipo Cuenta en la Base de Dados.

**Capa Dominio**

Se crea la clase **TypeAccount**, que contiene los atributos referentes a la clase tipo Cuenta de la Base de Datos.

**Capa Business**

Se crea la clase **TypeAccountBusiness** encargada de conectar la capa de datos con la capa de presentación.

**Capa Data.**

Se creó la clase **TypeAccountData** con la función **getTypeAccountData**, encargada de obtener todos los tipos de cuenta de la Base de Datos.

**Capa Presentación.**

Se modifica la clase **AccountInterface**, para que muestre los tipos de cuenta tanto al insertar como actualizar para las cuentas existentes.

* Se arregla la funcionalidad de ubicación y se implementa por capas. **Encargado Gustavo Nájera Nájera.**

**Capa Presentación**

Se modifica la clase **DirectionClient**, con el fin de que funcione, que haga llamado a la capa business en lugar de la capa Data y que se muestren las obciones seleccionadas antes, por ejemplo, si selecciona provincia Cartago, se mostrara el select de Provincia con Cartago seleccionado y apareceran los cantones de Cartago.

**Capa Business**

Se creó la clase **LocationBusiness**, la cual se encarga de conectar la capa de datos con la capa de presentación.

**Sistema de likes en muros:**

**Capa Data:** Se agregaron a la clase WallData algunos metodos adicionales que se encargan de la gestion de de los likes en los muros como lo son: getState(),updateLIke(),updatLikeChechked(),InsertNewLike(), los cuales se explicaran a continuacion.

EL metodo get state se encarga de obtener un registro de estado que coincide con el id del usuario que ha iniciado sesión y id del comentario. Esto para poder seleccionar cuales check han sido presionado como likes y cuales no.

Los metodos updateLIke() y updateLikeChechecked(), se encargan de actualizar en la base de datos cuando un comentario ha sido marcado o desmarcado como me gusta.

Por ultimo el metodo newLIke() se encarga de insertar un nuevo registro en la tabla cuando un usuario da por primera vez like a un comentario.

**Capa Business:** Esta capa contiene un objeto de la capa data para poder hacer un llamado de los metodos que han sido mencionados en la capa data, pero con la caracteristica que al final del nombre contienen el nombre de Business,sirve como punte entre la vista y el data.

**Capa Presentation:** En la capa Wall es donde se hace el llamado de los metodos que fueron mencionados en el business para dar la funcionalidad del me gusta a los comentarios, primero se hace un cargado de todos los comentarios, seguidamente se llama el metodo getStateBusiness() encargado de obtener los comentarios que han sido seleccionados como me gusta por un usuario, al ser marcado o desmarcado un like se invoca una funcion javascript checked() que es la encargada de llamar a los metodos updateLIkeChecked y updateLIke para cambiar el estado del me gusta.

• **Qué voy a hacer .**

* Mejorar script de predicción en búsquedas. **Encargado Gustavo Nájera Nájera**
* Almacenar la dirección al registrar la cuenta. **Encargado Gustavo Nájera Nájera**.
* Pasar búsquedas a la vista cliente. **Encargado Gustavo Nájera**
* Validaciones en cuenta del lado cliente. **Encargado Gustavo Nájera**
* Crear y utilizar tabla en BD tipoCuenta. **Encargado Gustavo Nájera**
* Se arregla la funcionalidad de ubicación y se implementa por capas. **Encargado Gustavo Nájera Nájera.**
* **I**mplementar el sistema de likes en los comentarios.Anthony Hernandez

• **Impedimentos**

* La funcionalidad de almacenar la dirección del dueño de la cuenta en cada una de sus cuentas se declara incompleta debido a que la funcionalidad de almacenar la ubicacion del cliente sigue incompleta **Encargado:Gustavo Nájera Nájera**, a la espera de que se concluya la funcionalidad de alamacenar la unbicacion de cada cliente **Encargado: Alberth Calderón Alvarado.**

• **Tareas realizadas que no fueron planeadas**.

* Se implementó una máscara para el número de cuenta. **Encargado: Gustavo Nájera Nájera.**

**2.** **Asuntos Pendientes**

· Tarea 1 que aún no se ha resuelto y siguen postergándose

· Tarea 2 que aún no se ha resuelto y siguen postergándose

· Tarea 3 que aún no se ha resuelto y siguen postergándose

**3.** **Asuntos Nuevos**

• Módulos funcionales nuevos que a la luz de las discusiones van siendo pertinentes de desarrollar, aún cuando no necesariamente lleguen a implementarse. No comprometen un hito de desarrollo

**4.** **Cumplimiento de tareas asignadas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Responsable** | **Grado de conclusión** | **Nota** |
| Mejorar script de predicción en búsquedas.  Pasar búsquedas a la vista cliente.  Validaciones en cuenta del lado cliente.  Crear y utilizar tabla en BD tipoCuenta.  Se arregla la funcionalidad de ubicación y se implementa por capas. | Gustavo Nájera Nájera | Concluido |  |
| Almacenar la dirección al registrar la cuenta. | Gustavo Nájera Nájera | Pendiente |  |
| Se agrega el sistema de likes al muro. | Anthony Hernandez Badilla | Concluido |  |
| Descripción de tarea 3 |  |  |  |

\*Aunque el proyecto es grupal, en cada iteración se asignará una persona del grupo como responsable de cada una de las tareas asignadas.

\*\*Existen tareas ya definidas como avances revisables en cada iteración:

1. Diagramas de casos de uso

2. Modelo de datos

3. Diagrama de aplicación

4. Módulos funcionales en ejecución

**PROXIMA REUNION:**

Fecha: 6 de febrero

Hora: 8:00 AM

Lugar: Laboratorio C, Sede del atlántico Turrialba

**CLAUSURA:**

Anthony Hernández Badilla como persona encargada de preceder la reunión, clausura la reunión a las 3pm horas.