UADY
"Luz, Ciencia y Verdad"

UNIVERSIDAD AUTÓNOMA DE YUCATÁN

Facultad de Matemáticas

Licenciatura en Ingeniería de Software

Ciclo Escolar Agosto-Diciembre 2023

Asignatura: Arquitectura de Software

CRUD con CodeIgneiter

Maestro:
Dr. Víctor Hugo Menéndez Domínguez

Integrantes:

- Jesús Oswaldo Chan Uicab
- Fernando Joachín Prieto
- Carlos Augusto May Vivas
- Reyna Valentina Ortiz Porras

Fecha de entrega: 06/Diciembre/2023

DESCRIPCIÓN DE LAS FUNCIONALIDADES

- **Módulo Models**

Objetivo Principal: Representa los modelos de Producto y Usuario para poder trabajar con la tabla correspondiente en la base de datos.

Clases involucradas: ProductoModel y Usuario Model.

Funcionalidades:

1. **password_encryptado():**
    - **Descripción:** Encriptar una contraseña del usuario.

- **Módulo Controllers**

Objetivo Principal: Administra las solicitudes para los diferentes servicios para la autenticación de un usuario y gestionar las operaciones CRUD con los productos.

Clases involucradas: AuthController y ProductoController.

Funcionalidades:

1. **crearUsuario():**
    - **Descripción:** Crea un usuario con las credenciales proporcionadas siempre que no exista en la base de datos, de lo contrario mostrará mensajes de error.
2. **iniciarSesion():**
    - **Descripción:** Crea la sesión del usuario si las credenciales proporcionadas son válidas, en caso de que no lo sean, muestra mensajes de error.
3. **logout():**
    - **Descripción:** Destruye la sesión actual del usuario y redirige a la página de inicio.
4. **mensaje():**
    - **Descripción:** Carga y devuelve la vista correspondiente a la página de mensaje la cual avisa que se creó la cuenta.
5. **vistaCrearUsuario():**
    - **Descripción:** Carga y devuelve la vista correspondiente a la página de crear usuario.

6. vistaIniciarSesion():

- **Descripción:** Carga y devuelve la vista correspondiente a la página de inicio de sesión.

7. crearProducto():

- **Descripción:** Procesa la creación de un nuevo producto, valida los datos del formulario, y, si son válidos, inserta el producto en la base de datos, de lo contrario muestra mensajes de error.

8. editarProducto():

- **Descripción:** Procesa la edición de un producto existente, valida los datos del formulario, y, si son válidos, actualiza el producto en la base de datos, de lo contrario muestra mensajes de error.

9. eliminarProducto():

- **Descripción:** Busca un producto por su ID, y si existe, lo elimina de la base de datos de forma permanente, de lo contrario, no realiza ninguna acción

10. vistaCrearProducto():

- **Descripción:** Carga y devuelve la vista correspondiente a la página de crear producto.

11. vistaEditarProducto():

- **Descripción:** Carga y devuelve la vista correspondiente a la página de editar producto.

12. vistaProducto():

- **Descripción:** Carga y devuelve la vista correspondiente a la página principal.

- **Módulo Entities**

Objetivo Principal: Representa la entidades de productos y usuarios con métodos la validación de cada una de las entidades.

Clases involucradas: ProductoEntity y UsuarioEntity

Funcionalidades:

1. **validar_producto():**
    - **Descripción:** Valida los datos de un producto, para luego retornar el arreglo de los errores cuyas validaciones no fueron superadas.
2. **comprobar_password():**
    - **Descripción:** Comprueba si la contraseña ingresada proporcionada por el usuario es la misma contraseña que tiene su cuenta en la base de datos.
3. **password_encryptado():**
    - **Descripción:** Encripta la contraseña del objeto de usuario para mejorar la seguridad de su cuenta.
4. **validar_cuenta():**
    - **Descripción:** Valida los datos de un producto, para luego retornar el arreglo de los errores cuyas validaciones no fueron superadas.
5. **validarLogin:**
    - **Descripción:** Valida los campos de email y contraseña para el proceso de inicio de sesión.
    -
5. **where():**
    - **Descripción:** Encuentra los registros de una consulta en la tabla de usuarios que tengan un valor específico en la columna proporcionada.

DESCRIPCIÓN DE COMPONENTES

1. Nombre del componente: ProductoModel

Descripción: Es un modelo que extiende la clase Model proporcionada por CodeIgniter. Este modelo está asociado con la tabla de base de datos llamada productos y define propiedades y configuraciones específicas para interactuar con esa tabla.

Dependencias con otros componentes: Este componente depende directamente del componente Model.

Interfaces de Salida: Se relacionan con la obtención y manipulación de datos en la tabla productos.

Interfaces de Entrada: Se relacionan con la forma en que los datos se proporcionan al modelo para su almacenamiento o manipulación, que serán útiles en el controlador ProductoController para las operaciones en la base de datos.

2. Nombre del componente: UsuarioModel

Descripción: Está asociado con la tabla de base de datos llamada usuarios y define propiedades y configuraciones específicas para interactuar con esa tabla.

Dependencias con otros componentes: Este componente depende directamente del componente Model.

Interfaces de Salida: Se relacionan con la obtención y manipulación de datos en la tabla de usuarios.

Interfaces de Entrada: Se relacionan con la forma en que los datos se proporcionan al modelo para su almacenamiento o manipulación, que serán útiles en el controlador AuthController para las operaciones en la base de datos.

3. Nombre del componente: AuthController

Descripción: Maneja la autenticación de usuarios y operaciones relacionadas con la creación y cierre de sesión.

Dependencias con otros componentes: El componente depende directamente de los modelos UsuarioModel para interactuar con la base de datos y UsuarioEntity para representar y validar los datos del usuario.

Además, depende del framework CodeIgniter ya que extiende la clase BaseController proporcionada por el framework.

También depende de vistas Login, Mensaje y Registro para renderizar las páginas de inicio de sesión, registro y mensajes.

Interfaces de Salida: Están relacionadas con las vistas que devuelve el controlador. En este caso, las vistas son "auth/login", "auth/registro" y "auth/mensaje". Además, el controlador redirige a otras páginas después de realizar ciertas acciones, como el inicio de sesión exitoso o la creación de un usuario.

Interfaces de Entrada: Están relacionadas con los datos que se obtienen de las solicitudes HTTP. El controlador utiliza estos datos para realizar operaciones como iniciar sesión (iniciarSesion), crear un usuario (crearUsuario), y realizar validaciones.

4. Nombre del componente: ProductoController

Descripción: Es un componente diseñado para representar productos en el contexto de una aplicación CodeIgniter, proporciona la funcionalidad para validar la integridad de los datos de un producto.

Dependencias con otros componentes: El componente depende directamente de los modelos ProductoModel para interactuar con la base de datos y ProductoEntity para representar y validar los datos del producto.

También depende del framework CodeIgniter ya que extiende la clase BaseController proporcionada por el framework.

Las vistas (archivos de plantilla) son parte de las dependencias, ya que el controlador las utiliza para generar la interfaz de usuario.

Interfaces de Salida: Están relacionadas con las vistas que devuelve el controlador. En este caso, las vistas son "producto/inicio", "producto/crear", y "producto/editar". Además, el controlador redirige a otras páginas después de realizar ciertas acciones, como la creación o edición de un producto.

Interfaces de Entrada: Están relacionadas con los datos que se obtienen de las solicitudes HTTP. Por ejemplo, al crear o editar un producto, el controlador utiliza los datos enviados a través del método POST para realizar las operaciones correspondientes.

Artefactos:

- **Archivos:**

  datatable.js: Se está utilizando jQuery para extender las opciones predeterminadas de DataTables y personalizar la apariencia y el comportamiento de las tablas generadas por este plugin.

  node.js: Ejecuta código de JavaScript del lado del servidor.

5. Nombre del componente: ProductoEntity

Descripción: Encapsula la lógica y validaciones relacionadas con los datos de productos, específicamente la validación de los campos "nombre", "precio" y "disponibles".

Dependencias con otros componentes: Este componente depende directamente del componente Entities.

Interfaces de Salida: Están relacionadas con las vistas que devuelve el controlador. En este caso, las vistas son "producto/inicio", "producto/crear", y "producto/editar". Además, el controlador redirige a otras páginas después de realizar ciertas acciones, como la creación o edición de un producto.

Interfaces de Entrada: Están relacionadas con los datos que se obtienen de las solicitudes HTTP. Por ejemplo, al crear o editar un producto, el controlador utiliza los datos enviados a través del método POST para realizar las operaciones correspondientes.

6. Nombre del componente: UsuarioEntity

Descripción: Encapsula la lógica y validaciones relacionadas con los datos de usuario, como la validación de inicio de sesión, la validación de creación de cuenta, encriptación de contraseña, comprobación de contraseña y una función para realizar consultas a la base de datos.

Dependencias con otros componentes: Este componente depende directamente del componente Entities.

Interfaces de Salida: Están relacionadas con el retorno de alertas en caso de errores de validación.

Interfaces de Entrada: Están relacionadas con los datos que se pasan a través de la función validar_producto(). La función espera datos de la entidad para realizar las validaciones.

7. Nombre del componente: Alertas

Descripción: Muestra alertas en una interfaz de usuario.

Dependencias con otros componentes: Este componente depende directamente del componente Template.

Interfaces de Salida: Envía respuestas HTTP para mostrar las alertas según sea el caso.

Interfaces de Entrada: Solicitudes HTTP que indican el tipo de alerta.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

8. Nombre del componente: LayoutAuth

Descripción: Vista de un formulario para que el usuario inicie sesión, cree un nuevo usuario y restablezca la contraseña en caso de ser olvidada.

Dependencias con otros componentes: Este componente depende directamente del componente Template.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

9. Nombre del componente: Layout

Descripción: Vista de un formulario para ver los detalles y manipular los datos de un producto, ya sea editar, actualizar o eliminar.

Dependencias con otros componentes: Este componente depende directamente del componente Template.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

10. Nombre del componente: Login

Descripción: Vista de un formulario para que el usuario inicie sesión.

Dependencias con otros componentes: Este componente depende del componente Alertas para visuañizar los errores si no se llegó a cumplir correctamente con los campos solicitados.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

11. Nombre del componente: Mensaje

Descripción: Vista donde confirma si al crearse un nuevo usuario se hizo de manera correcta.

Dependencias con otros componentes: Este componente depende del componente Alertas para visualizar los mensajes de éxito si se llegó a cumplir correctamente con los campos solicitados.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

12. Nombre del componente: Registro

Descripción: Vista de un formulario para que el usuario pueda crearse una cuenta.

Dependencias con otros componentes: Este componente depende del componente Alertas para visualizar los errores si no se llegó a cumplir correctamente con los campos solicitados.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

13. Nombre del componente: Inicio

Descripción: Vista que forma parte del menú principal para poder crear, eliminar, editar y ver los detalles de los productos.

Dependencias con otros componentes: Este componente depende directamente del componente Producto.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

14. Nombre del componente: Crear

Descripción: Vista que proporciona los datos necesarios para poder crear un nuevo producto.

Dependencias con otros componentes: Este componente depende del componente Alertas para visualizar los errores si no se cumplió correctamente con los campos solicitados.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

15. Nombre del componente: Editar

Descripción: Vista que proporciona los datos necesarios para poder editar un producto que se encuentre almacenado en la base de datos.

Dependencias con otros componentes: Este componente depende del componente Alertas para visualizar los errores si no se cumplió correctamente con los campos solicitados.

Interfaces de Salida: Envía respuestas HTTP, proporcionando datos y confirmación de operaciones. Las respuestas pueden incluir información para ser mostrada en la interfaz de usuario y detalles sobre el resultado de las operaciones realizadas.

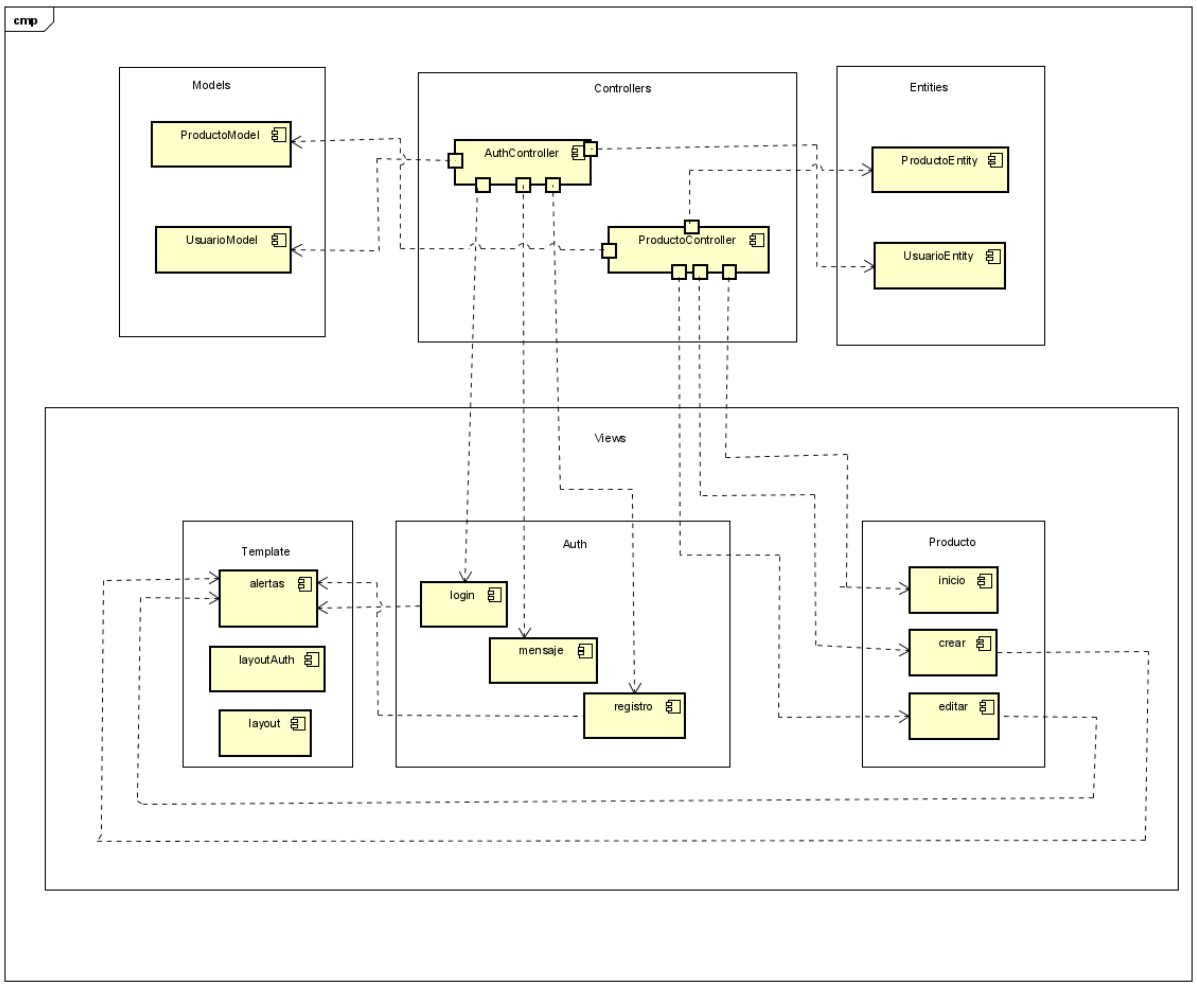
Interfaces de Entrada: Puede recibir solicitudes HTTP de diversos tipos, como GET, POST, PUT y DELETE. Estas solicitudes actúan como mecanismos clave para la interacción entre el usuario y la aplicación, permitiendo que el frontend envíe información al backend y solicite diversas operaciones.

Artefactos:

- **Archivos:**

  node.js: Ejecuta código de JavaScript del lado del servidor.

DIAGRAMA DE COMPONENTES UML



DESCRIPCIÓN DE CLASES

Nombre de la clase: AuthController

Descripción: La clase AuthController representa un controlador destinado a manejar la autenticación de usuarios. Contiene métodos para gestionar el inicio de sesión, cierre de sesión y creación de nuevos usuarios.

Dependencias con otras clases:

- **UsuarioEntity**
    - **Asociación:** Creación de una instancia de UsuarioEntity.
    - **Descripción:** Se utiliza para representar y gestionar la entidad de usuario. La instancia $auth se crea a partir de esta clase y se utiliza para recopilar datos relacionados con el inicio de sesión y la validación.

- **UsurarioModel**
    - **Asociación:** Creación de una instancia de UsuarioModel.
    - **Descripción:** Se utiliza para interactuar con la base de datos en relación con la gestión de usuarios. Se instancia en los métodos crear y logout para realizar operaciones de inserción y búsqueda.

Atributos:

- **$request**
    - **Tipo:** Mixed
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** No se proporciona un valor por omisión ya que se inicializa en el constructor.
    - **Descripción:** Instancia del objeto principal de solicitud.

- **$helpers**
    - **Tipo:** Array
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** No se proporciona un valor por omisión.
    - **Descripción:** Un array que contiene los nombres de los helpers que se cargarán automáticamente.

Funciones:

- **vistaIniciarSesion()**
    - **Argumentos:** No hay argumentos en la función.

    - **Tipo de Retorno:** La función devuelve un objeto de tipo view.
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** Devuelve una vista.
    - **Componente e Interface de Salida:** No aplica en este caso, ya que la función no implementa un servicio específico.
    - **Descripción:** Esta función devuelve una vista que representa la interfaz de inicio de sesión **(auth/login)**.

- **vistaCrearUsuario()**
    - **Argumentos:** No hay argumentos en la función.
    - **Tipo de Retorno:** La función devuelve un objeto de tipo view.
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** Devuelve una vista.
    - **Componente e Interface de Salida:** No aplica en este caso, ya que la función no implementa un servicio específico.
    - **Descripción:** Esta función devuelve una vista que representa la interfaz de registro de usuario (auth/registro).

- **mensaje():**
    - **Argumentos:** No hay argumentos en la función.
    - **Tipo de Retorno:** La función devuelve un objeto de tipo view.
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** Devuelve una vista.
    - **Descripción:** Esta función devuelve una vista que representa la interfaz de registro de usuario (auth/mensaje).

- **iniciarSesion():**
    - **Argumentos:** No hay argumentos definidos en la función, pero la función utiliza propiedades o métodos de la instancia actual ($this->request).
    - **Tipo de Retorno:** La función puede devolver diferentes tipos de valores:
        - Si las credenciales son válidas, devuelve una redirección (redirect()).
        - Si hay errores, devuelve una redirección a la página de inicio con datos de entrada y errores (redirect()->to(base_url())->withInput()->with('errors', $alertas)).
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** Está relacionada con la autenticación de usuarios y la gestión de sesiones.
    - **Descripción:** Esta función realiza el proceso de inicio de sesión. Obtiene las credenciales del formulario de inicio de sesión, las valida utilizando un método validarLogin() en la clase UsuarioEntity, y si las credenciales son válidas, establece una sesión y redirige al usuario a la página de inicio. Si hay errores, redirige a la página de inicio de sesión con los errores y datos de entrada para informar al usuario sobre los problemas encontrados durante el

intento de inicio de sesión. Además, maneja el caso en el que el usuario no exista en la base de datos.

- **crearUsuario():**
    - **Argumentos:** No hay argumentos definidos en la función, pero la función utiliza propiedades o métodos de la instancia actual ($this->request).
    - **Tipo de Retorno: L**a función puede devolver diferentes tipos de valores:
        - Si se crea con éxito un nuevo usuario, devuelve una redirección (redirect()->to(base_url('/mensaje'))).
        - Si hay errores durante el proceso, devuelve una redirección a la página de registro con datos de entrada y errores (redirect()->to(base_url('registro'))->withInput()->with('errors', $alertas)).
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** Está relacionada con la creación de **u**suarios y la validación de cuentas.
    - **Descripción:** Esta función maneja la lógica de creación de usuarios. Obtiene los datos del formulario de registro, crea una instancia de la clase UsuarioEntity para representar al nuevo usuario, valida la cuenta con el método validar_cuenta(), y si la cuenta es válida y el usuario no está registrado previamente, intenta insertar el nuevo usuario en la base de datos. Si la inserción es exitosa, redirige al usuario a una página de mensaje. En caso de errores, redirige a la página de registro con los errores y datos de entrada para informar al usuario sobre los problemas encontrados durante el intento de creación de usuario.

- **logout():**
    - **Argumentos:** No hay argumentos en la función.
    - **Tipo de Retorno:** La función devuelve una redirección (redirect()->to(base_url('/'))).
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** Está relacionada con la acción de cerrar sesión (logout).
    - **Descripción:** Esta función, llamada logout, realiza la acción de cerrar sesión en la aplicación web. Primero, destruye la sesión actual utilizando session()->destroy(). Luego, redirige al usuario a la página de inicio (base_url('/')). En resumen, la función se encarga de cerrar la sesión del usuario y redirigirlo a la página principal de la aplicación.
- **Atributos**: No hay atributos de clase explícitos en la clase.

Nombre de la clase: ProductoController

Descripción: Controlador para gestionar las operaciones relacionadas con productos.

Dependencias con otras clases:

- **ProductoEntity:** Entidad utilizada para representar los datos de un producto.
- **ProductoModel:** Modelo que interactúa con la base de datos para realizar operaciones relacionadas con productos.

Atributos: No tiene atributos de clase explícitos en la clase.

Funciones:

- **vistaProducto():**
    - **Argumentos:** No hay argumentos en la función.
    - **Tipo de Retorno:** La función devuelve una vista que representa la interfaz de inicio de la sección de productos.
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** La función está relacionada con la obtención de datos de productos y la renderización de una vista.
    - **Componente e interface de salida:** La función devuelve una vista que está asociada a la interfaz de inicio de productos ("producto/inicio").
    - **Descripción:** La función se encarga de obtener la información de todos los productos y renderizar la vista de inicio de productos, para mostrar un catálogo de productos.

- **vistaCrearProducto():**
    - **Argumentos:** No hay argumentos en la función.
    - **Tipo de Retorno:** La función devuelve un objeto de tipo view.
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** Devuelve una vista.
    - **Descripción:** Esta función devuelve una vista que representa la interfaz para crear un producto (producto/crear).

- **vistaEditarProducto():**
    - **Argumentos:** La función tiene un argumento llamado $id con un tipo entero.
    - **Tipo de Retorno:** La función devuelve una vista que representa la interfaz de edición de un producto.
    - **Visibilidad:** Pública
    - **Implementación de Servicio:** La función está relacionada con la obtención de datos de un producto para su edición y la renderización de una vista.
    - **Descripción:** La función se encarga de obtener la información de un producto específico por su ID y renderizar la vista de edición de productos, para permitir al usuario modificar la información de un producto.

- **crearProducto():**
  - **Argumentos:** No hay argumentos en la función.
  - **Tipo de Retorno:** La función puede devolver diferentes tipos de valores:
    - **Si se crea con éxito un nuevo producto**, devuelve una redirección (redirect()->to(base_url('/inicio'))).
    - **Si hay errores durante el proceso**, devuelve una redirección a la página de creación de productos con datos de entrada y errores (redirect()->to(base_url('inicio/producto/crear'))->withInput()->with('errors', $alertas)).
  - **Visibilidad:** Pública
  - **Implementación de Servicio:** La función está relacionada con la creación de productos y la validación de productos.
  - **Descripción:** la función implementa la funcionalidad de creación de productos, validación de productos y gestión de errores para cuando no se cumple algún campo al momento que se desea crear un producto.

- **editarProducto():**
  - **Argumentos:** La función tiene un argumento llamado $id con un tipo entero.
  - **Tipo de Retorno: La función puede devolver diferentes tipos de valores:**
    - **Si la edición es exitosa**, devuelve una redirección (redirect()->to(base_url('/inicio'))).
    - **Si hay errores durante el proceso**, devuelve una redirección a la página de edición de productos con datos de entrada y errores (redirect()->to(base_url('inicio/producto/editar/'.$id))->withInput()->with('errors', $alertas)).
  - **Visibilidad:** Pública
  - **Implementación de Servicio:** La función está relacionada con la edición de productos y la validación de productos.
  - **Descripción:** Esta función implementa la funcionalidad de edición de productos, validación de productos y gestión de errores para cuando no se cumple algún campo al momento que se desea editar un producto.

- **eliminarProducto():**
  - **Argumentos:** La función tiene un argumento llamado $id con un tipo entero.
  - **Tipo de Retorno:** Si la eliminación es exitosa, devuelve una redirección (redirect()->to(base_url('/inicio'))).
  - **Visibilidad:** Pública
  - **Implementación de Servicio:** La función está relacionada con la eliminación de productos.
  - **Descripción:** La función implementa la funcionalidad de eliminación de productos, tomando como entrada el ID del producto a eliminar.

Nombre de la clase: ProductoEntity

Descripción: Es una entidad diseñada para representar productos en el contexto de una aplicación CodeIgniter, proporciona la funcionalidad para validar la integridad de los datos de un producto.

Dependencias con otras clases: Depende de la clase base ***'Entity'*** proporcionada por CodeIgniter para que pueda extender funcionalidades de la clase base ***'Entity'*** y así trabajar como una entidad.

Atributos:

- **$alertas**
    - **Visibilidad:** Protected
    - **Valor por omisión:** Null []
    - **Descripción:** Atributo estático utilizado para almacenar mensajes de alerta generados durante la validación

Funciones:

- **validar_producto():**
    - **Argumentos:** Ninguno
    - **Valor de retorno:** Array
    - **Visibilidad:** Pública
    - **Servicios implementados:** Este método no implementa una interfaz específica. Su función principal es realizar validaciones en la entidad de producto.
    - **Descripción:** Realiza la validación de los atributos de la entidad Producto los cuales son ***$nombre***, ***$precio*** y ***$disponibles***, en dado caso de que ninguno de los atributos no estuviera presente, se agrega mensajes de error al array ***$alertas*** y retorna dicho array.

Nombre de la Clase: UsuarioEntity

Descripción: Representa una entidad de usuario en una aplicación CodeIgniter, contiene métodos para la validación de datos relacionados con el usuario, manipulación de contraseñas y consultas a la base de datos.

Dependencias con otras clases: Depende de la clase base ***'Entity'*** proporcionada por CodeIgniter para que pueda extender funcionalidades de la clase base ***'Entity'*** y así trabajar como una entidad.

Atributos:

- **$alertas**
    - **Tipo:** Array
    - **Visibilidad:** Protected

    - **Valor por Omisión:** Null []
    - **Descripción:** Atributo estático utilizado para almacenar mensajes de alerta generados durante la validación.

Funciones:

- **validarLogin():**
    - **Argumentos:** Ninguno
    - **Valor de Retorno:** Array
    - **Visibilidad:** Pública
    - **Servicios Implementados:** Este método no implementa una interfaz específica. Su función principal es realizar validaciones en la entidad de la cuenta del usuario.
    - **Descripción:** Realiza la validación del email y la contraseña del usuario. Agrega mensajes de error al array ***$alertas*** si los campos no cumplen con las condiciones especificadas.

- **validar_cuenta():**
    - **Argumentos:** Ninguno
    - **Valor de Retorno:** Array
    - **Visibilidad:** Pública
    - **Servicios Implementados:** Este método no implementa una interfaz específica. Su función principal es realizar validaciones en la entidad de usuario.
    - **Descripción:** Realiza la validación de los atributos nombre, apellido, email, y password de la entidad UsuarioEntity. Agrega mensajes de error al array estático $alertas si los campos no cumplen con las condiciones especificadas.

- **password_encryptado():**
    - **Argumentos:** Ninguno
    - **Valor de Retorno:** Ninguno
    - **Visibilidad:** Pública
    - **Servicios Implementados:** Este método no implementa una interfaz específica. Su función principal es encriptar una contraseña
    - **Descripción:** Encripta la contraseña actual de la entidad utilizando el algoritmo Bcrypt.

- **comprobar_password():**
    - **Argumentos:**
        - $passwordComprobar (Tipo: String) - Contraseña a encriptar
    - **Valor de Retorno:** Boolean
    - **Visibilidad:** Pública

    - **Servicios Implementados:** Este método no implementa una interfaz específica. Su función principal es comparar dos contraseñas
    - **Descripción:** Comprueba si la contraseña proporcionada coincide con la contraseña almacenada en la entidad.

- **where():**
    - **Argumentos:**
        - $columna (Tipo: String) - Columna que hace referencia a un atributo de la tabla usuario
        - $valor (Tipo: mixed) - Valor que debe tener el registro a encontrar
    - **Valor de Retorno:** Array
    - **Visibilidad:** Pública
    - **Servicios Implementados:** Este método no implementa una interfaz específica. Su función principal es realizar una consulta en la base de datos
    - **Descripción:** Realiza una consulta a la base de datos para encontrar registros de la tabla 'usuarios' donde la columna especificada coincide con el valor proporcionado.

Nombre de la clase: UsuarioModel

Descripción: Representa el modelo asociado a la tabla de usuarios en una base de datos. Utiliza el framework CodeIgniter y extiende la clase Model. Esta clase define la estructura y comportamiento del modelo para interactuar con la tabla de usuarios.

Dependencias con otras clases: Extiende la clase base Model proporcionada por CodeIgniter, lo que significa que hereda y utiliza sus funcionalidades para interactuar con la base de datos.

Atributos:

- **$table:**
    - **Tipo:** String
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** usuarios
    - **Descripción:** Nombre de la tabla de la base de datos asociada al modelo. La tabla se llama usuarios.

- **$primaryKey:**
    - **Tipo:** String
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** id
    - **Descripción:** Nombre del campo que sirve como clave primaria en la tabla de la base de datos. La clave primaria es el campo id.

- **$useAutoIncrement:**
    - **Tipo:** Boolean
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** true
    - **Descripción:** Indica si se utiliza la autoincrementación para la clave primaria.

- **$returnType:**
    - **Tipo:** String
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** array
    - **Descripción:** Tipo de datos que se espera devolver de las operaciones de consulta, en donde se espera un array.

- **$useSoftDeletes:**
    - **Tipo:** Boolean
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** false
    - **Descripción:** Indica si se utiliza eliminación suave (soft deletes), donde los registros no se eliminan físicamente sino que se marcan como eliminados, en donde no se utiliza eliminación suave (false).

- **$allowedFields:**
    - **Tipo:** Array
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** ['nombre', 'apellido', 'email', 'password']
    - **Descripción:** Lista de campos permitidos para operaciones de base de datos, en donde se permiten los campos nombre, apellido, email y password.

- **$beforeInsert:**
    - **Tipo:** Array
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** ["password_encryptado"]
    - **Descripción:** Lista de métodos a ejecutar antes de realizar una inserción en la base de datos, en dondese ejecuta el método password_encryptado.

Funciones:

- **validarLogin():**
    - **Argumentos:** La función tiene un argumento llamado $data de tipo array.
    - **Valor de Retorno:** Array
    - **Visibilidad:** Pública

- **Servicios Implementados:** La función está relacionada con la manipulación de datos antes de realizar una inserción en la base de datos.
- **Descripción:** Implementa la funcionalidad de cifrar contraseñas antes de realizar una inserción en la base de datos.

Nombre de la clase: ProductoModel

Descripción: Es responsable de la interacción con la tabla de productos en una base de datos, definiendo cómo se manejan los datos y cómo interactuar con la base de datos mediante la extensión de la clase base Model de CodeIgniter.

Dependencias con otras clases: Extiende la clase base Model proporcionada por CodeIgniter. Esto implica que hereda funcionalidades de la clase base para interactuar con la base de datos, realizar consultas y manipular datos.

Atributos:

- **$allowedFields:**
  - **Tipo:** array <string>
  - **Visibilidad:** Protegida
  - **Valor por Omisión:** [‘nombre’, ‘precio’, ‘disponibles’]
  - **Descripción:** Incluye las columnas de la tabla del modelo correspondiente en la base de datos.

- **$primaryKey:**
  - **Tipo:** String
  - **Visibilidad:** Protegida
  - **Valor por Omisión:** id
  - **Descripción:** Hace referencia al nombre de la columna que sirve como llave primaria en la tabla de la base de datos.

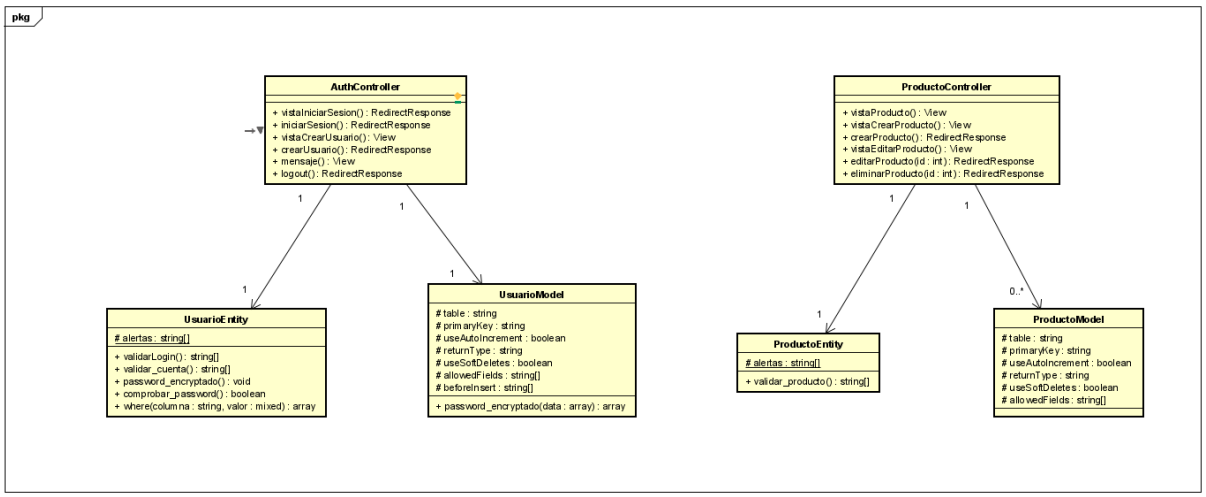
- **$useAutoIncrement:**
  - **Tipo:** Boolean
  - **Visibilidad:** Protegida
  - **Valor por Omisión:** true
  - **Descripción:** Indica si se utiliza la autoincrementación para la clave primaria.

- **$returnType:**
  - **Tipo:** String
  - **Visibilidad:** Protegida
  - **Valor por Omisión:** object
  - **Descripción:** Indica el tipo de retorno esperado al realizar consultas.

- **$useSoftDeletes:**
    - **Tipo:** Boolean
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** false
    - **Descripción:** IIndica si se utiliza la eliminación suave en el modelo.

- **$table:**
    - **Tipo:** string
    - **Visibilidad:** Protegida
    - **Valor por Omisión:** ‘productos’
    - **Descripción:** Hace referencia a la tabla de la base de datos asociada a este modelo.

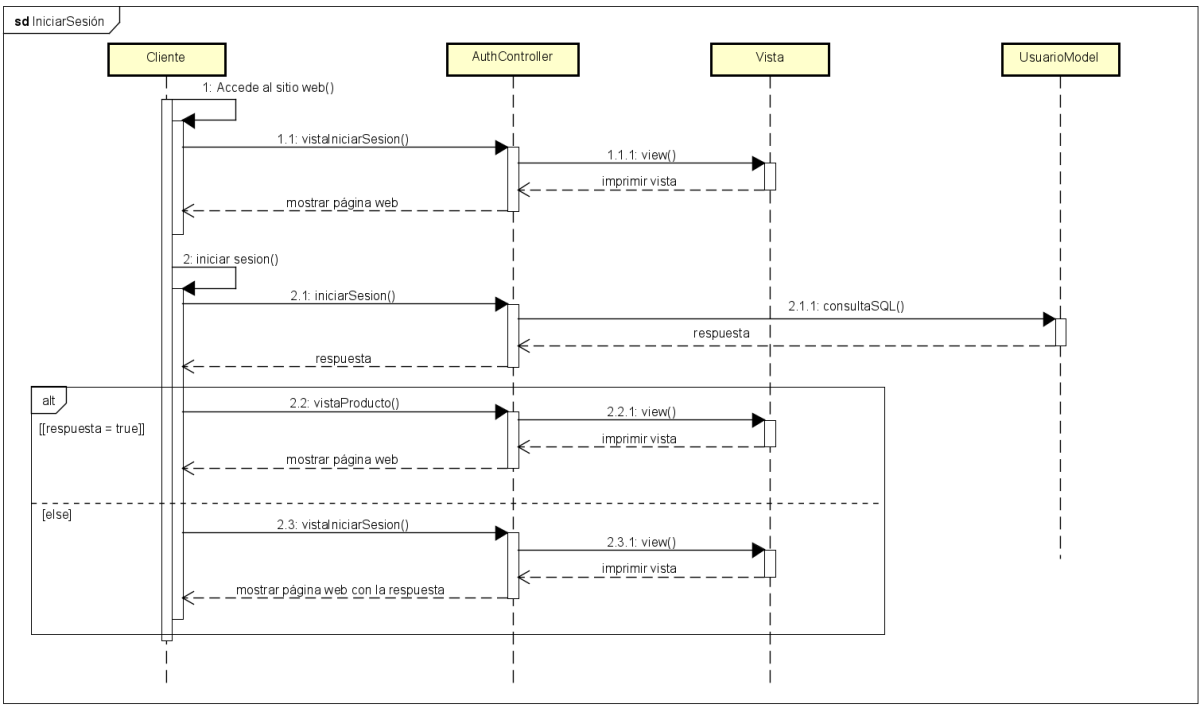
DIAGRAMA DE CLASES UML



DESCRIPCIÓN DE LAS SECUENCIAS

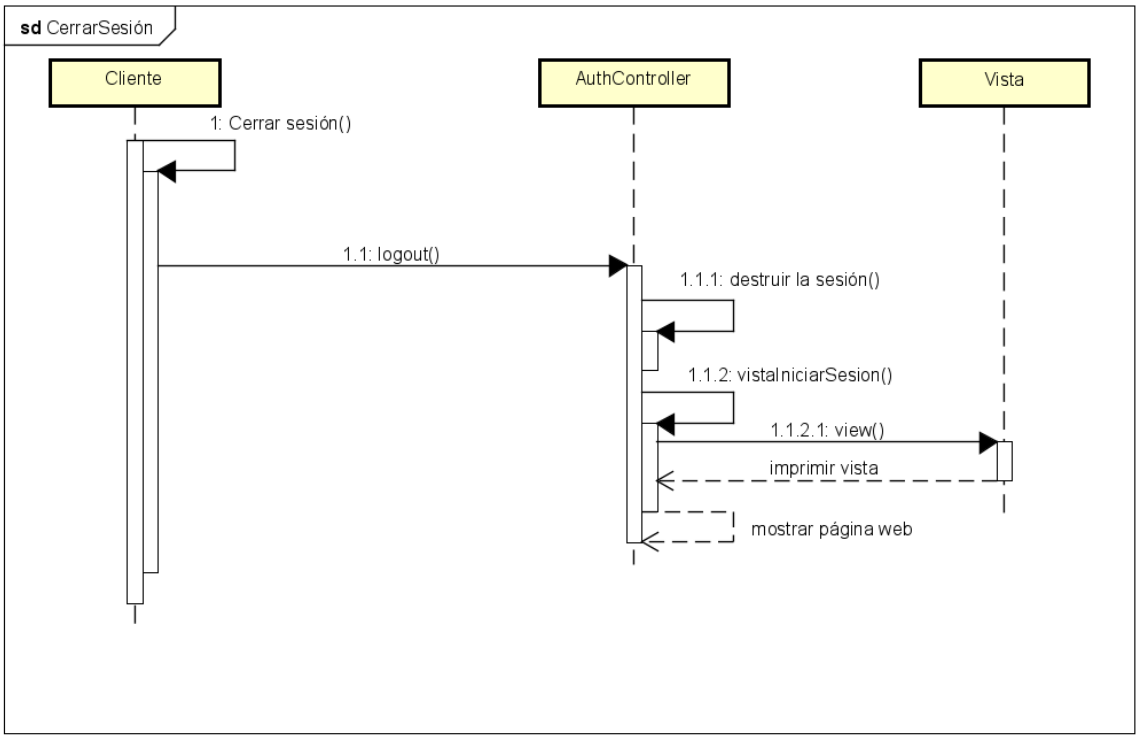
Nombre: Iniciar Sesión

Descripción: Secuencia para que un usuario inicie sesión y se verifique que ese usuario esté en la base de datos, para poder validar su acceso o no.



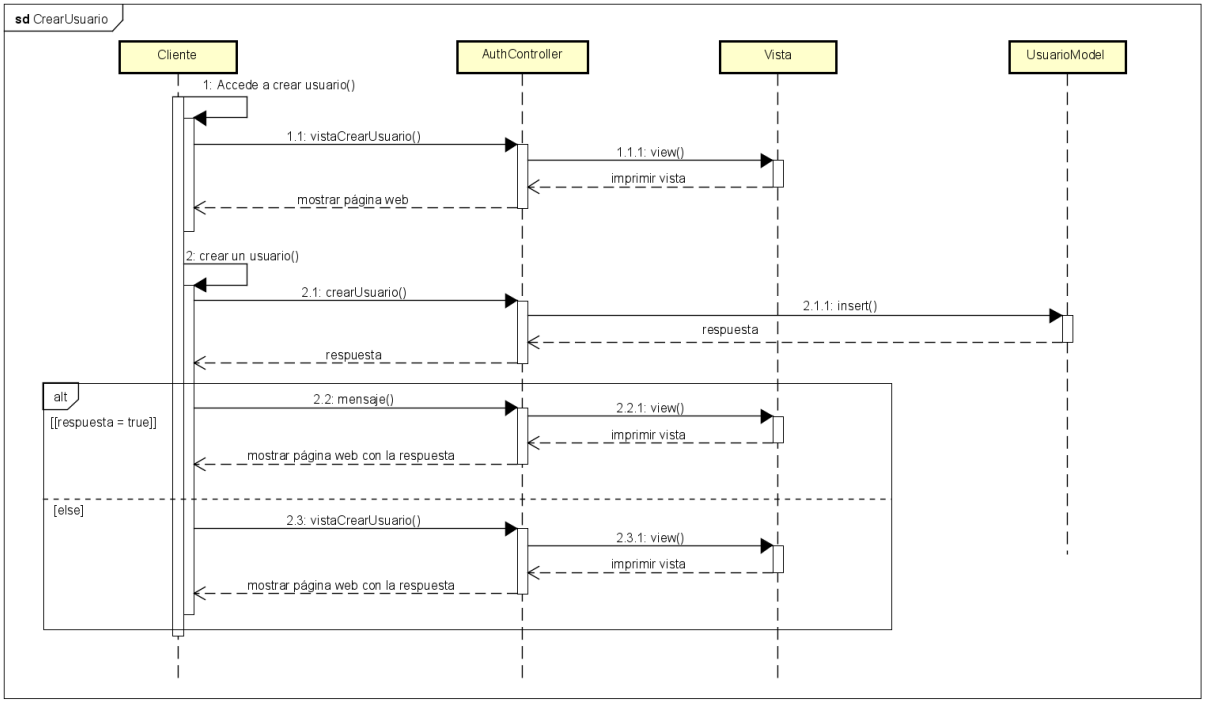
Nombre: Cerrar Sesión

Descripción: Secuencia para cuando un usuario cierre sesión de su cuenta y lo renderiza a la página de inicio de sesión.



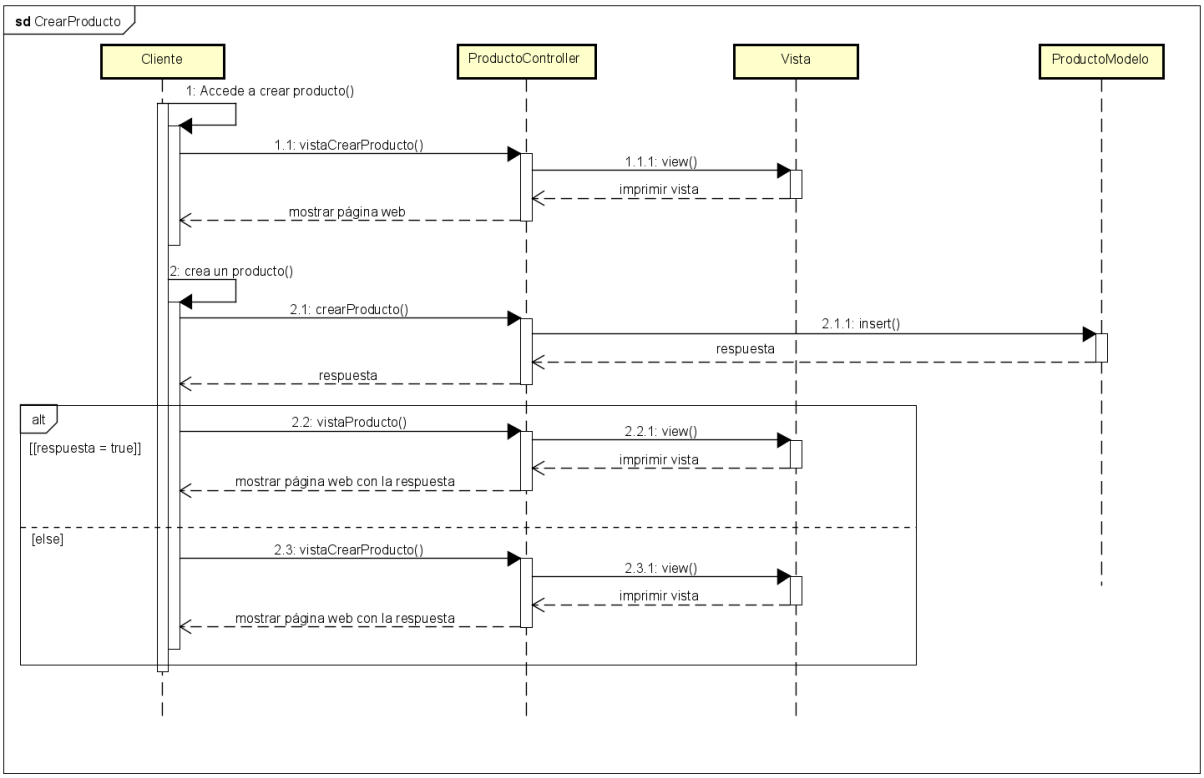
Nombre: Crear Usuario

Descripción: Secuencia para cuando se quiere dar de alta a un usuario en la base de datos, donde valida si los datos ingresados son correctos y renderiza a una vista según sea el caso.



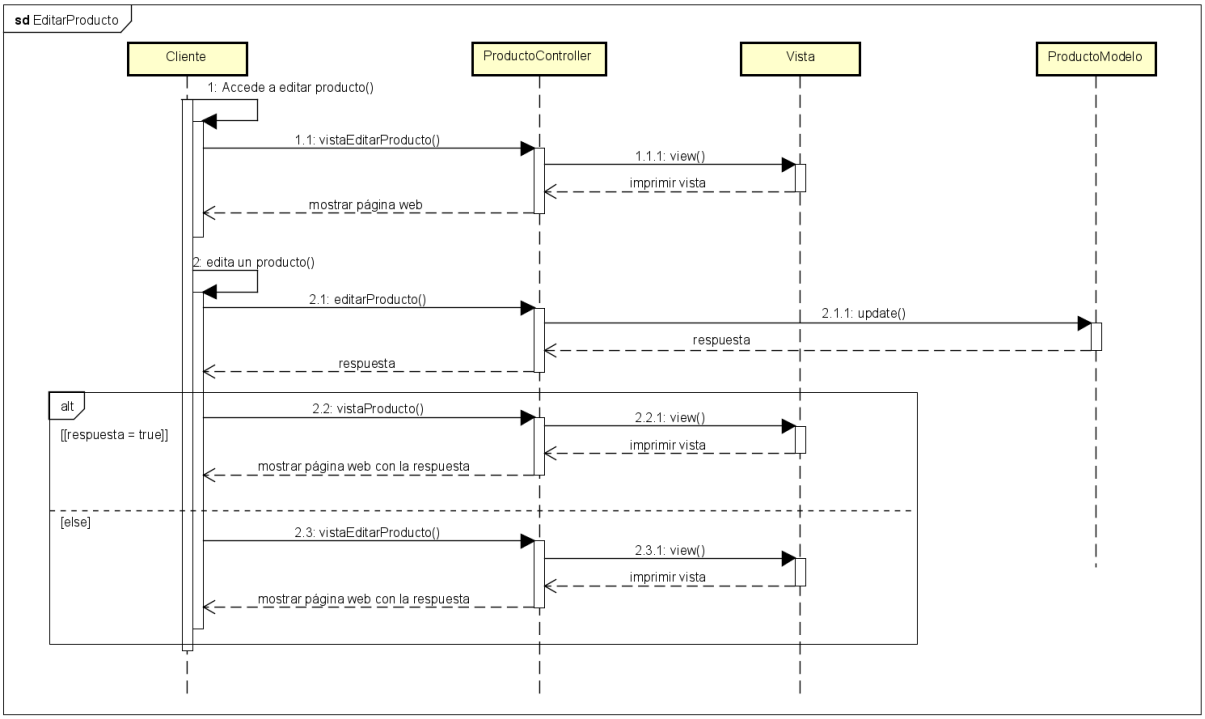
Nombre: Crear Producto

Descripción: Secuencia de la creación de un producto, actualizando estos cambios en la base de datos.



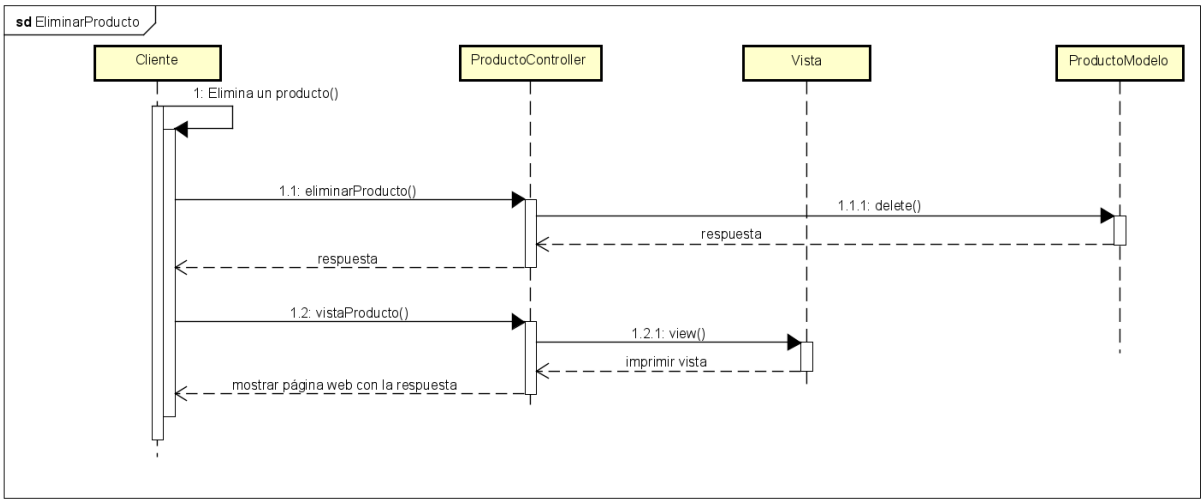
Nombre: Editar Producto

Descripción: Secuencia para editar un producto con las especificaciones del usuario, actualizando estos cambios en la base de datos.



Nombre: Eliminar Producto

Descripción: Secuencia para eliminar un producto, actualizando estos cambios en la base de datos.



Link del video: https://www.youtube.com/watch?v=VmBZr9DLbEs