Proyecto Análisis y Diseño de Sistemas de Información
Sistema Pokédex

Adrián Ramírez Gómez, Maira Herbas Jaldin, Miguel García Jiménez, Urko Méndez Espiga

17 de enero de 2026

Índice general

Capítulo 1

Casos de uso

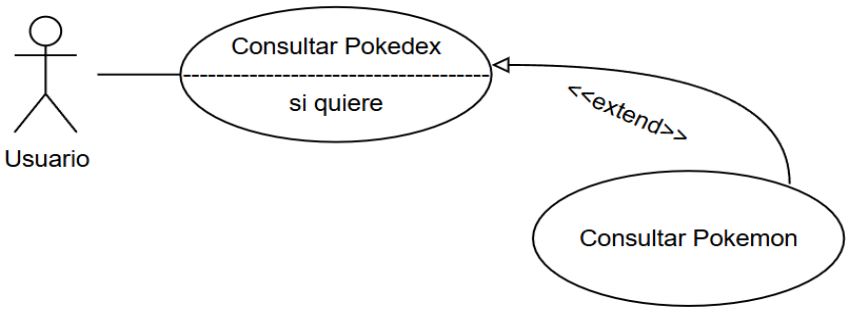

Figura 1.1: Diagrama Consultar Pokemon

1.1. Consultar Pokemon

- **Nombre**. Consultar Pokemon.
- **Descripción**. Permite al usuario consultar los datos Pokemon almacenados en la Pokédex, tales como, su foto, estadísticas y una breve descripción.
- **Actores** . Usuario.
- **Precondiciones**.
  1. El usuario ha iniciado sesión.
  2. El usuario está en la ventana de .ªquipos Pokémon.º "Pokédex".
  3. Ha seleccionado un Pokémon de la lista.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El usuario selecciona el pokemon del cual desea obtener información
  2. Se le muestra el nombre del pokemon junto con su imagen, estadísticas y descripción.
- **Postcondiciones**.

1. Éxito: El usuario visualiza los datos del Pokémon.
2. Error: Se muestra un mensaje y se conserva el estado anterior.

Interfaz Gráfica

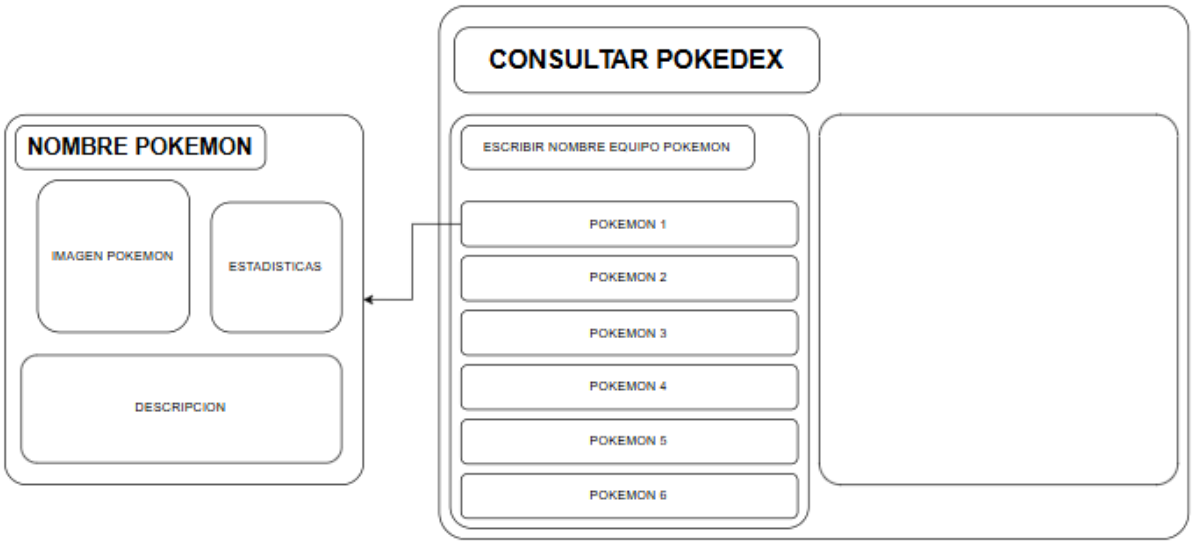

Figura 1.2: Interfaz Consultar Pokemon.

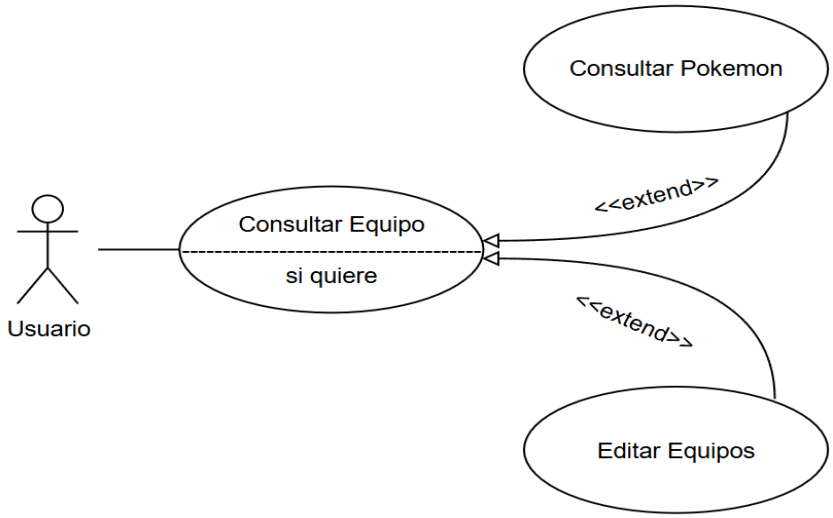

Figura 1.3: Diagrama Consultar Equipo

1.2. Consultar Equipo.

- **Nombre**. Consultar Equipo.
- **Descripción**. Permite ver los pokemons que están en un equipo, el nombre del equipo y si se quiere poder editar los equipos, ver los detalles de un pokemon o eliminar algún equipo.
- **Actores** . Usuario.
- **Precondiciones**.
  1. El usuario ha iniciado sesión en la aplicación.
  2. El usuario se encuentra en el menú principal.
  3. El usuario tiene al menos un equipo Pokémon creado.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El usuario selecciona el equipo del que quiere recibir información.
  2. Se muestra la información del equipo
     a) Si se quiere se puede consultar un pokemon del equipo y obtener sus datos
     b) Si se quiere se puede editar el equipo
     c) Si se quiere se puede eliminar el equipo actual
- **Postcondiciones**.
  1. Éxito: El usuario visualiza su equipo Pokémon en una ventana nueva.
  2. Error: Se muestra mensaje de error y se regresa al menú principal.

Interfaz Gráfica

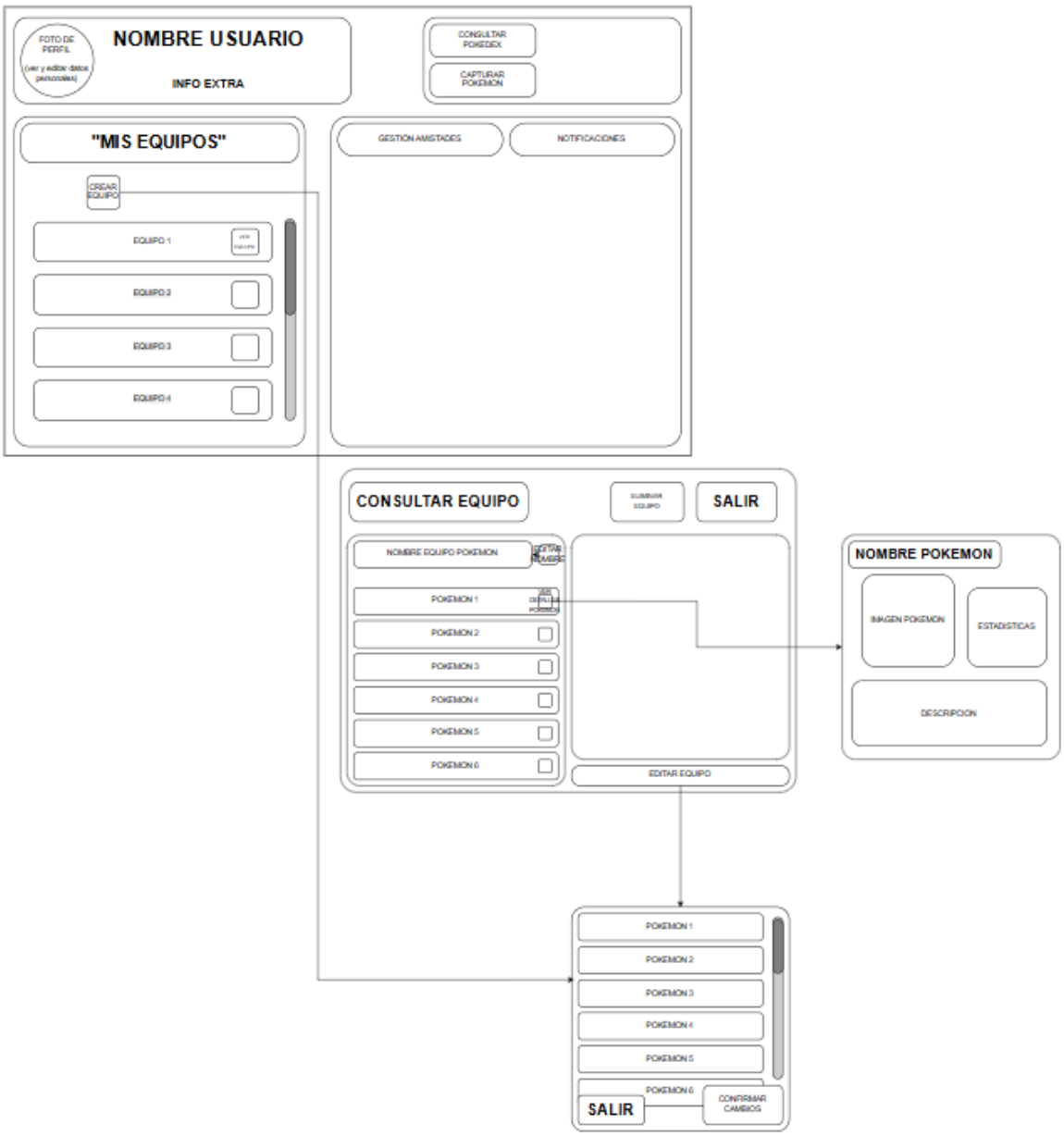

Figura 1.4: Interfaz Consultar Equipo

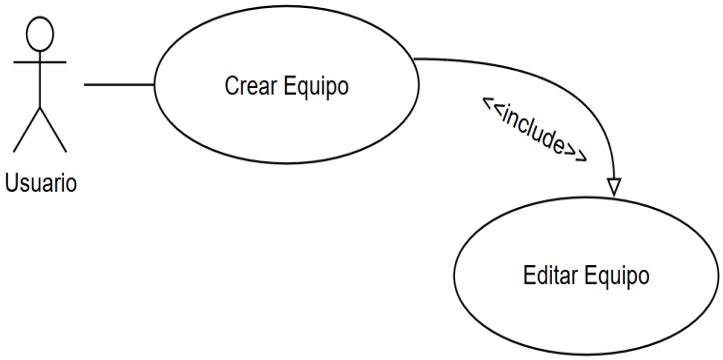

Figura 1.5: Diagrama Crear Equipo

1.3. Crear Equipo.

- **Nombre**. Crear Equipo.
- **Descripción**. Es un botón al que al pulsarlo se abre una nueva ventana en donde puedes añadir los pokemons que desees añadir a un equipo.
- **Actores** . Usuario.
- **Precondiciones**.
  1. El usuario ha iniciado sesión en la aplicación.
  2. El usuario se encuentra en el menú principal.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El usuario selecciona la opción de crear equipo.
  2. Se abre la pestaña de editar equipo desde la que selecciona 6 pokemon o menos.
     a) Se puede confirmar el equipo.
     b) Se puede cancelar la acción.
- **Postcondiciones**.
  1. Éxito: Nuevo equipo creado y visible en el menú principal.
  2. Cancelación: No se crea equipo y se regresa al estado anterior.
  3. Error: Se muestra mensaje y se regresa sin cambios.

Interfaz Gráfica

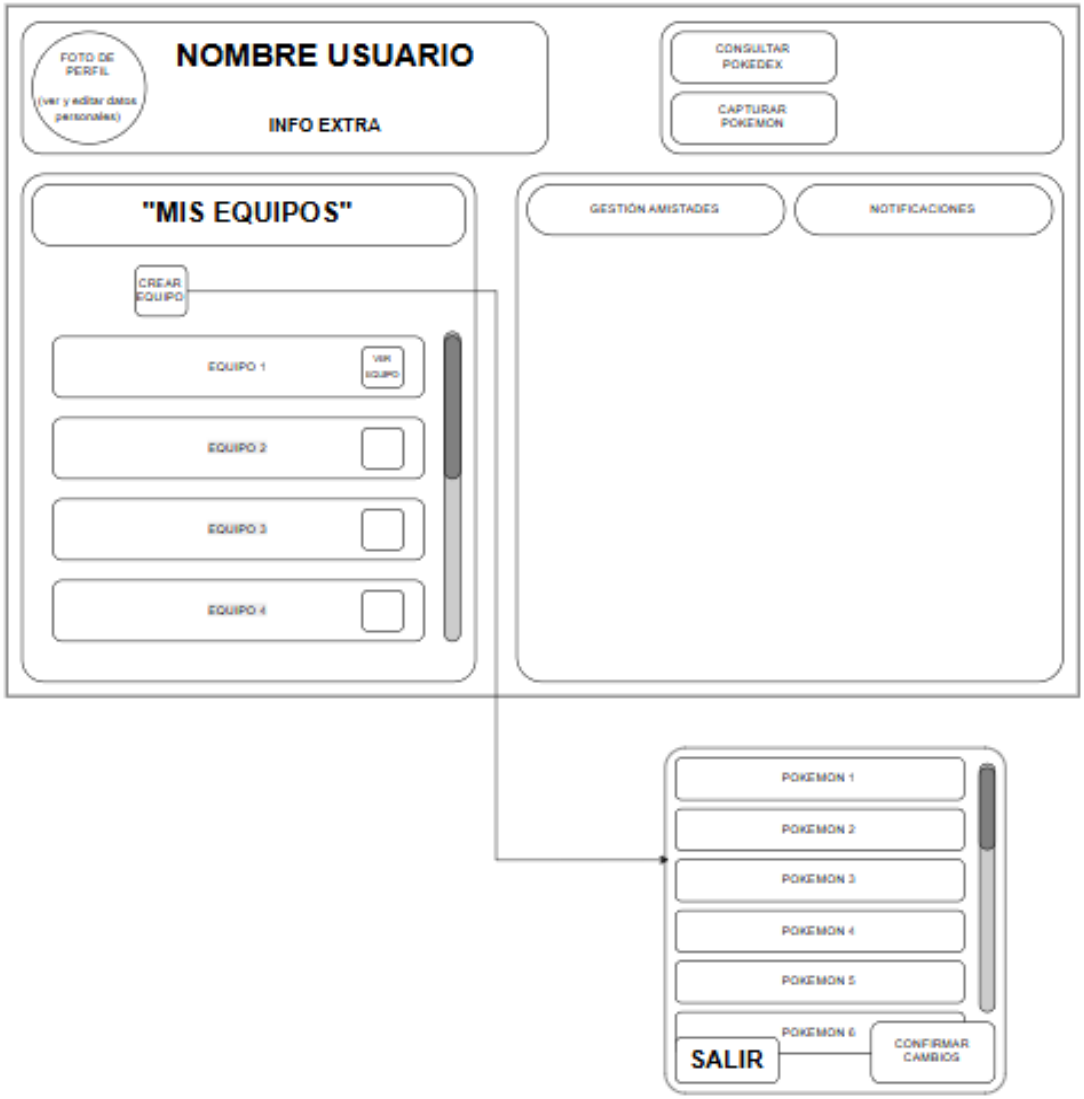

Figura 1.6: Interfaz Crear Equipo

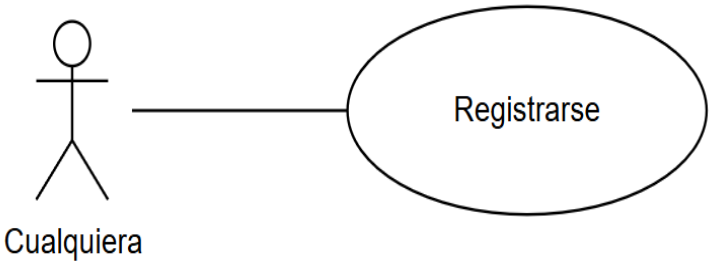

Figura 1.7: Diagrama Registrarse

1.4. Registrarse

- **Nombre**. Registrarse.
- **Descripción**. Permite al usuario crear una cuenta para poder acceder al sistema.
- **Actores** . Cualquiera.
- **Precondiciones**.
  1. El usuario accede a la aplicación desde la pantalla de login.
  2. El usuario tiene los datos necesarios para completar el formulario.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El usuario selecciona la opción de Registrarse.
  2. Se lleva a una pestaña donde introduce sus datos.
     a) Se puede confirmar una vez todos los datos sean introducidos.
     b) Se puede salir.
- **Postcondiciones**.
  1. Éxito: Cuenta creada en estado "pendiente de aprobación". Usuario no puede iniciar sesión hasta ser aprobado por administrador.
  2. Error: Se mantiene en pantalla de registro con mensajes de error.
  3. Cancelación: Regresa a pantalla de inicio sin cambios.

Interfaz Gráfica

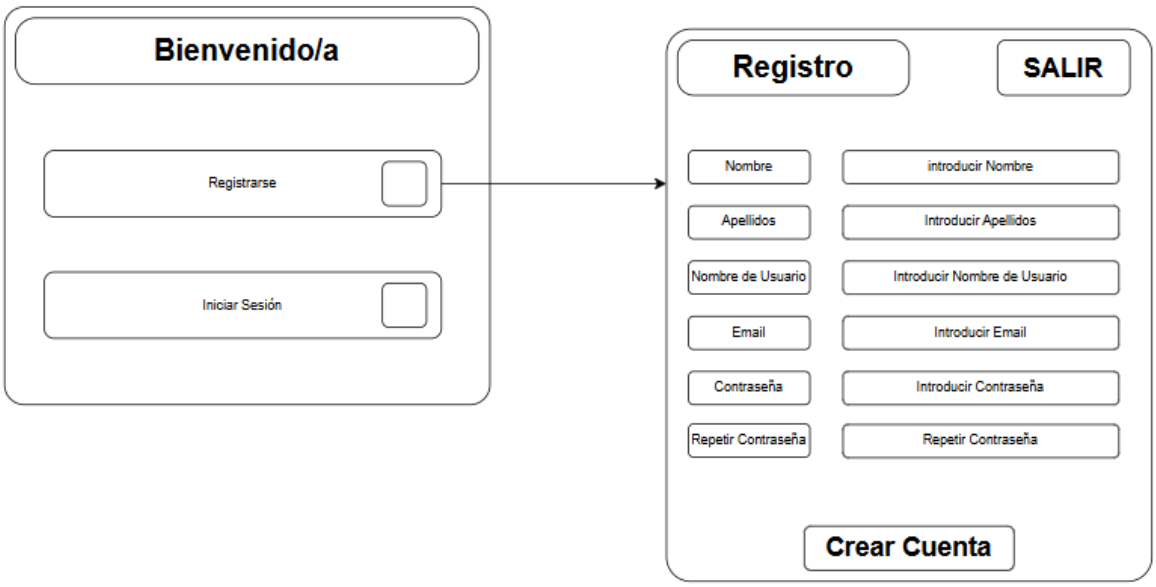

Figura 1.8: Interfaz Registrarse

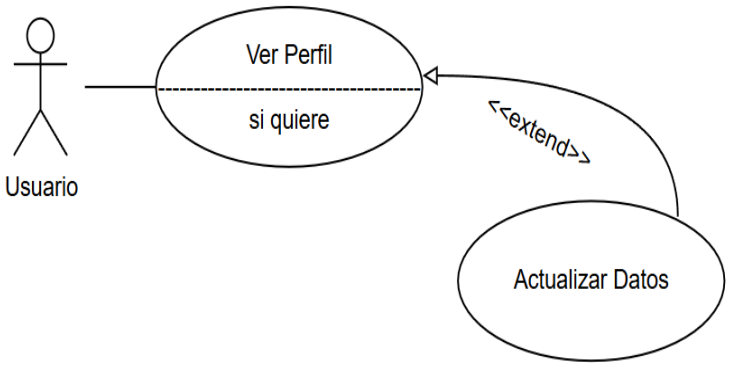

Figura 1.9: Diagrama Ver Perfil

1.5. Ver Perfil

- **Nombre**. Ver Perfil.
- **Descripción**. Permite al usuario actualizar sus datos personales en caso de ser necesario, dentro de la ventana de “MI PERFIL” .
- **Actores** . Usuario.
- **Precondiciones**.
  1. El usuario ha iniciado sesión.
  2. El usuario está en la ventana "Menú principal".
  3. El dato a editar está visible y disponible para modificación.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El usuario selecciona su icono de perfil.
  2. Se muestra una pestaña con su información.
     a) Si quiere puede entrar a editar sus datos.
     b) Se da la opción de volver al menú principal.
- **Postcondiciones**.
  1. Éxito: El dato modificado se guarda y muestra actualizado.
  2. Cancelación: El dato mantiene su valor original.
  3. Error: Se muestra mensaje y se conserva el valor anterior.

Interfaz Gráfica

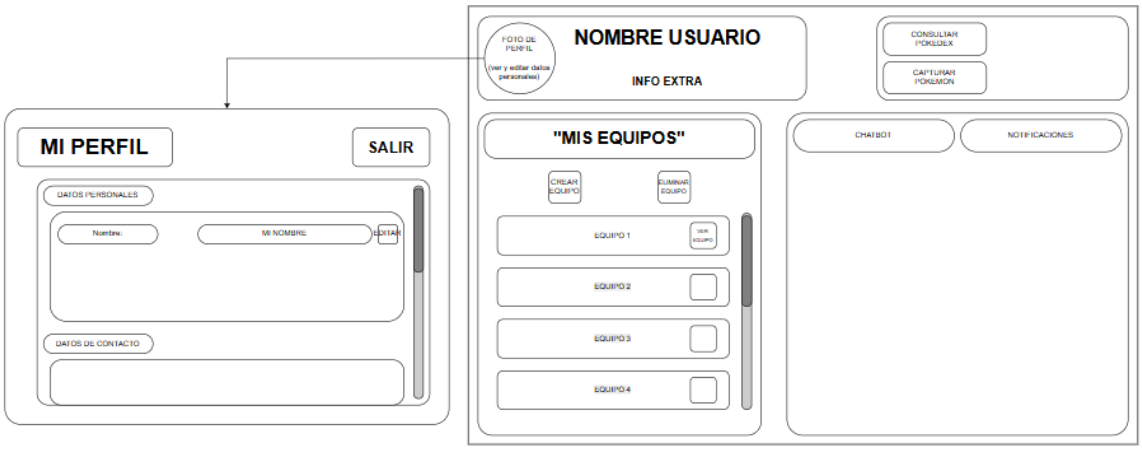

Figura 1.10: Interfaz Ver Perfil

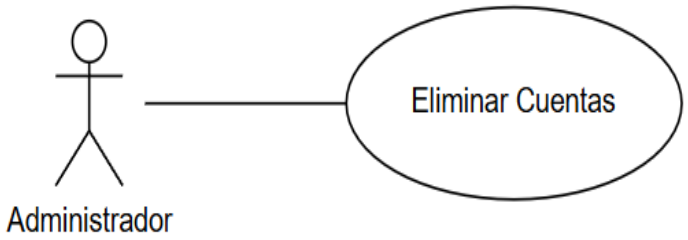

Figura 1.11: Diagrama Eliminar Cuentas

1.6. Eliminar Cuentas

- **Nombre**. Eliminar Cuentas.
- **Descripción**. En caso de que el administrador quiera eliminar alguna cuenta puede hacerlo mediante éste botón dentro del menú principal del administrador.
- **Actores** . Administrador.
- **Precondiciones**.
  1. El administrador ha iniciado sesión con credenciales de admin.
  2. El administrador se encuentra en el menú principal de administración.
  3. Existen cuentas de usuario en el sistema (excluyendo la cuenta del admin).
  4. La lista de usuarios está visible en el menú principal.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El administrador selección el botón de eliminar desde el usuario que quiera eliminar.
     a) Se puede confirmar la acción.
     b) Se puede cancelar la acción.
- **Postcondiciones**.
  1. Éxito: Las cuentas seleccionadas son eliminadas y la UI vuelve al estado normal.
  2. Parcial: Algunas cuentas se eliminan y otras fallan.
  3. Cancelación: La UI vuelve al estado normal sin cambios en los usuarios.
  4. Error: Se muestra mensaje y se mantienen todas las cuentas.

Interfaz Gráfica

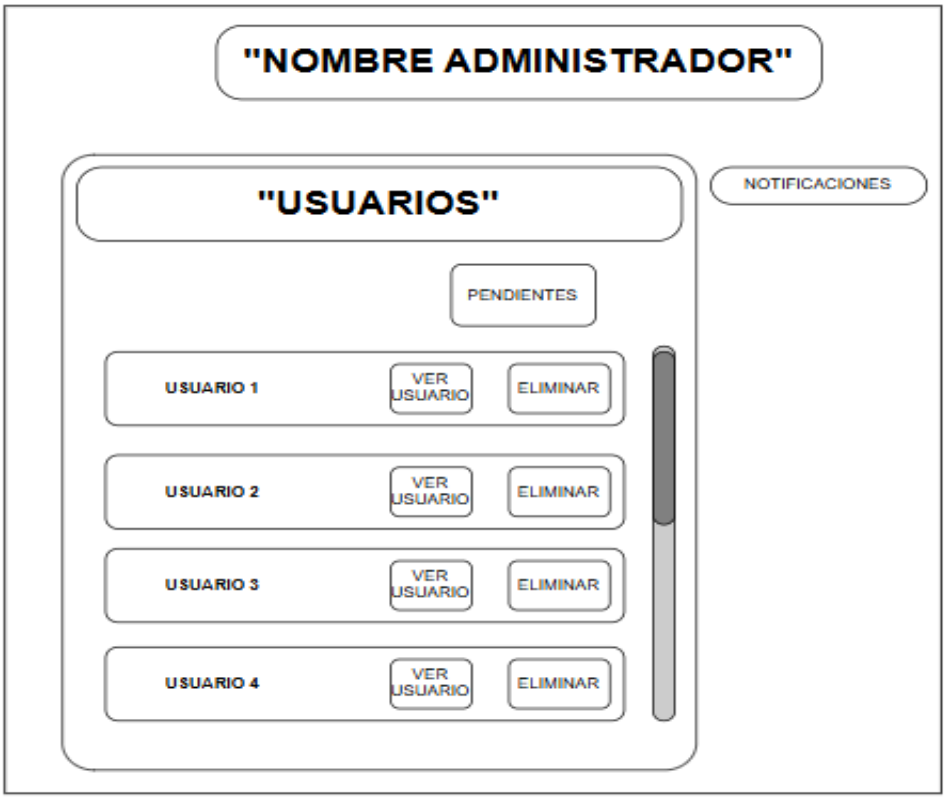

Figura 1.12: Interfaz Eliminar Cuentas

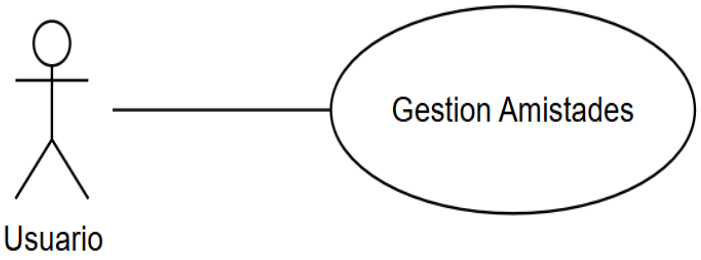

Figura 1.13: Diagrama Gestión Amistades

1.7. Gestión Amistades.

- **Nombre**. Gestión Amistades.
- **Descripción**. Función que se usa para poder consultar las amistades, ver las solicitudes pendientes y enviar solicitudes de amistad, todo esto, dentro de una ventana nueva.
- **Actores** . Usuario.
- **Precondiciones**. Estar identificado en el sistema.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**

  1. El usuario selecciona el boton de gestionar amistades.
  2. Se le muestra una pantalla donde puede ejecutar una de las isguientes acciones:
     a) Consultar sus amistades.
     b) Consultar las solicitudes pendientes.
     c) enviar una solicitud de amistad.

- **Postcondiciones**.

  1. Éxito: Se aceptan solicitudes y se tienen nuevas amistades.
  2. Extensión: Aparece una nueva notificación en el perfil del amigo que se ha enviado solicitud.
  3. Error: Se muestra mensaje de error y opción de volver al menú principal.

Interfaz Gráfica

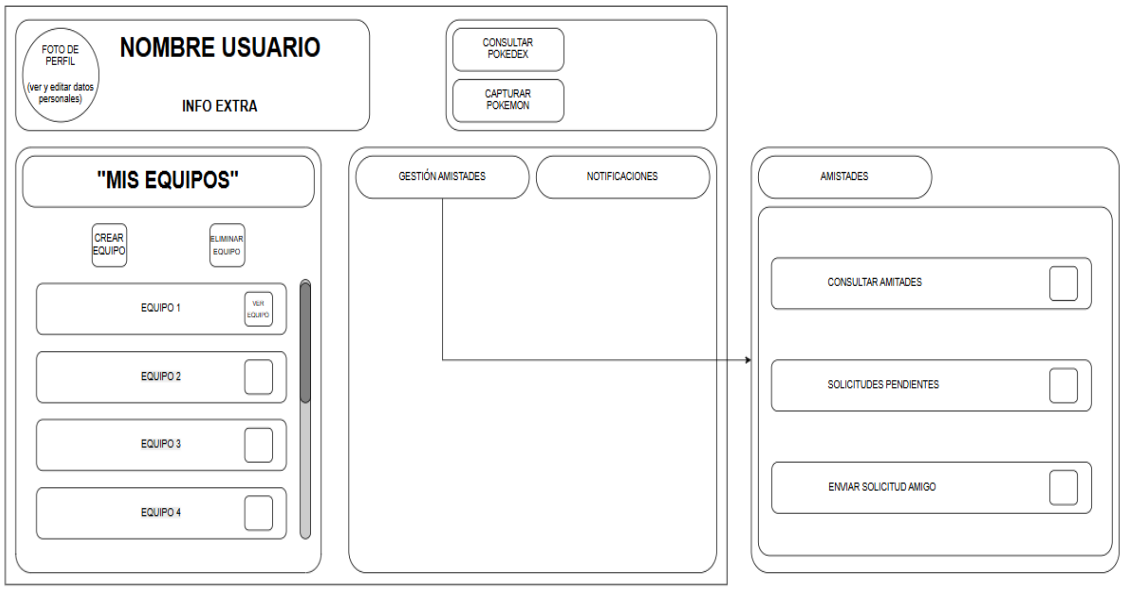

Figura 1.14: Interfaz Gestión Amistades

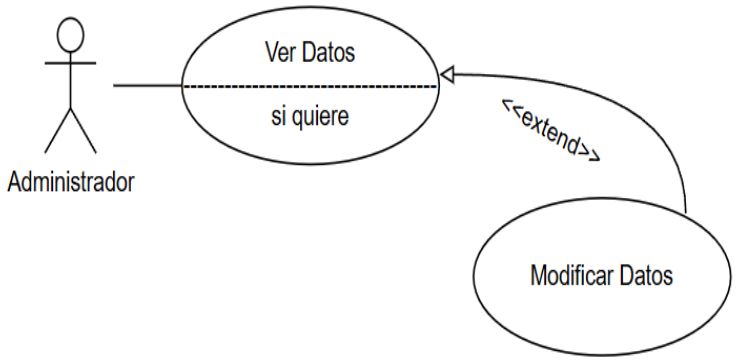

Figura 1.15: Diagrama Ver Datos

1.8. Ver Datos

- **Nombre**. Ver datos.
- **Descripción**. Dentro de una una nueva interfaz, permite ver los datos asociados a un usuario ya aprobado, y modificar algún dato en caso de ser necesario.
- **Actores** . Administrador.
- **Precondiciones**.
  1. El administrador ha iniciado sesión en la aplicación.
  2. El administrador se encuentra en su menú principal.
  3. Los datos de los usuarios están disponibles en el sistema.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El administrador selecciona un usuario.
  2. Se muestra una pantalla con los diferentes datos del usuario.
     a) El administrador puede modificar los datos del usuario si así lo desea.
- **Postcondiciones**.
  1. Éxito: El usuario visualiza y navega por la Pokédex correctamente.
  2. Extensión: Puede acceder a detalles específicos de cualquier Pokémon.
  3. Error: Se muestra mensaje de error y opción de volver al menú principal.

Interfaz Gráfica

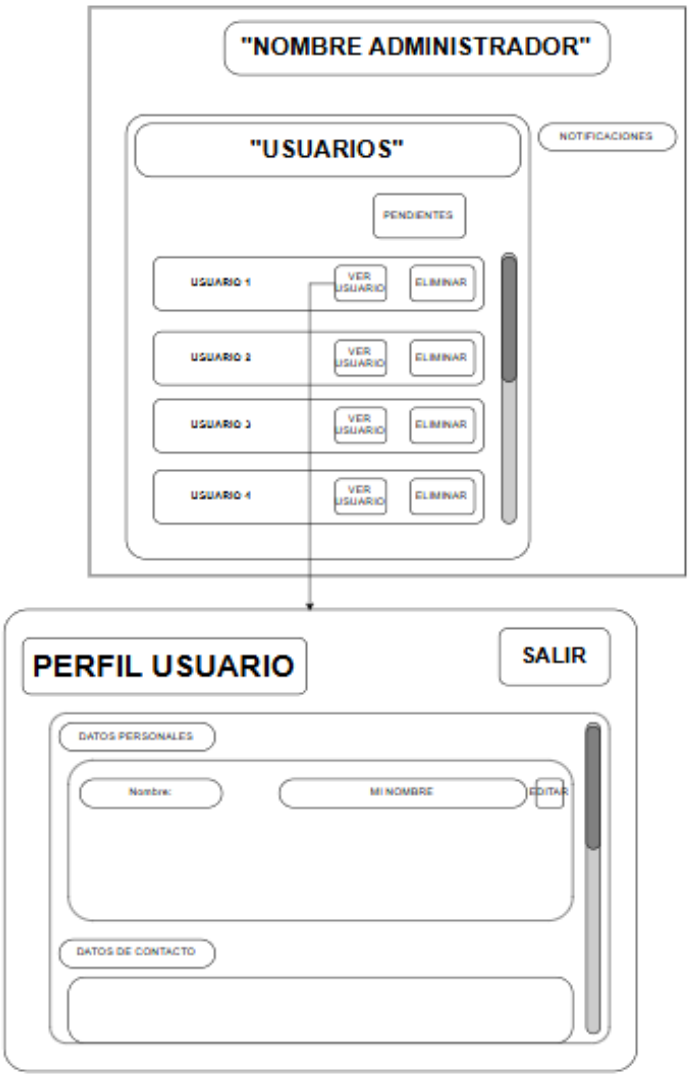

Figura 1.16: Interfaz Ver Datos

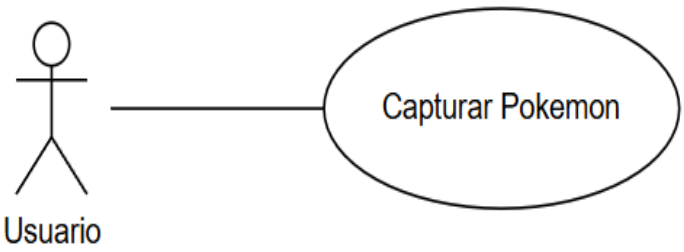

Figura 1.17: Diagrama Capturar Pokemon

1.9. Capturar Pokemon

- **Nombre**. Capturar Pokemon
- **Descripción**. Nos lleva a la interfaz gráfica de una ruleta en donde te puede tocar un pokémon aleatorio.
- **Actores** . Usuario.
- **Precondiciones**. Estar identificado en el sistema.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El usuario selecciona la opción Capturar Pokemon.
  2. Se muestra en pantalla una pestaña donde aparece una ruleta que te selecciona un pokemon de forma aleatoria.
  3. El pokemon se añade a la lista de pokemon del usuario.
- **Postcondiciones**.
  1. Éxito: Se suma el pokemon a la lista del usuario
  2. Error: Se muestra mensaje de error y opción de reintentar.

Interfaz Gráfica

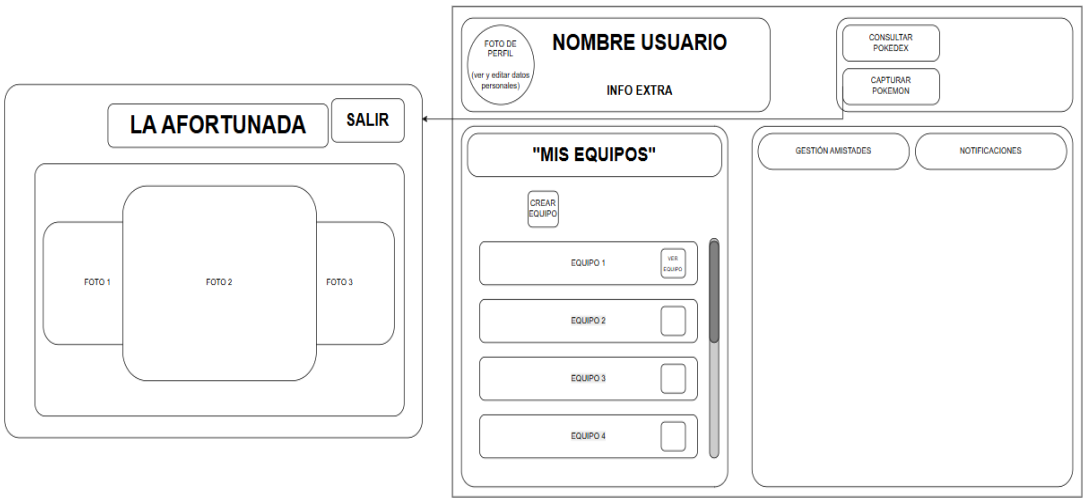

Figura 1.18: Interfaz Capturar Pokemon

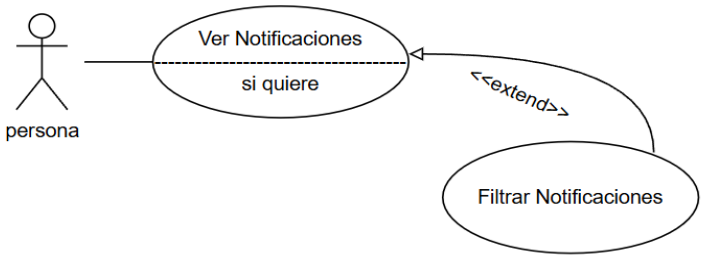

Figura 1.19: Diagrama Ver Notificaciones

1.10. Ver notificaciones.

- **Nombre**. Ver notificaciones.
- **Descripción**. Una nueva pestaña que nos muestra el historial de notificaciones, y el botón para poder filtrar notificaciones en caso de ser necesario.
- **Actores** . Persona
- **Precondiciones**.
  1. El actor ha iniciado sesión en la aplicación.
  2. El actor se encuentra en el menú principal.
  3. La bandeja de notificaciones está disponible.
  4. El actor está viendo la bandeja de notificaciones.
  5. Existen notificaciones disponibles para filtrar.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. La persona selecciona la opción de ver sus notificaciones.
     a) La persona puede filtrar las notificaciones si así lo desea.
- **Postcondiciones**.
  1. Éxito: La bandeja muestra notificaciones filtradas.
  2. Sin resultados: Muestra mensaje "No hay notificaciones con estos filtros".
  3. Limpieza: Al quitar filtros, se muestran todas las notificaciones.
  4. Error: Se muestra mensaje de error y opción de reintentar.

Interfaz Gráfica

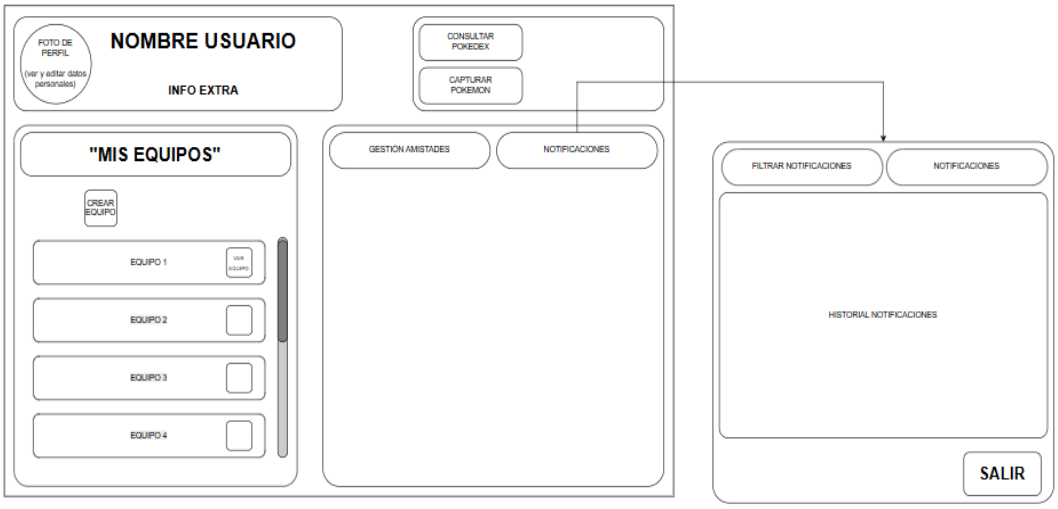

Figura 1.20: Interfaz Ver Notificaciones

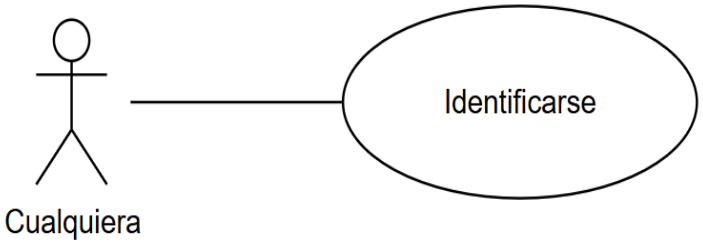

Figura 1.21: Diagrama Identificarse

1.11. Identificarse

- **Nombre**. Identificarse.
- **Descripción**. Cualquier usuario ya registrado, podrá identificarse con su nombre de usuario y contraseña. Si son correctos nos llevará al menú principal.
- **Actores** . Cualquiera.
- **Precondiciones**.
  1. El usuario ha iniciado sesión en la aplicación.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. Cualquiera selecciona la opción de identificarse al abrir la aplicación.
  2. Se muestra por pantalla una pestaña donde se pueden insertar los datos para iniciar sesión.
  3. Tras añadir los datos cualquiera selecciona la pestaña de iniciar sesión.

  Postcondiciones.
  1. Éxito: El usuario accede a su perfil
  2. Error: Se muestra la pantalla de identificarse de nuevo y se muestra un mensaje de error.

Interfaz Gráfica

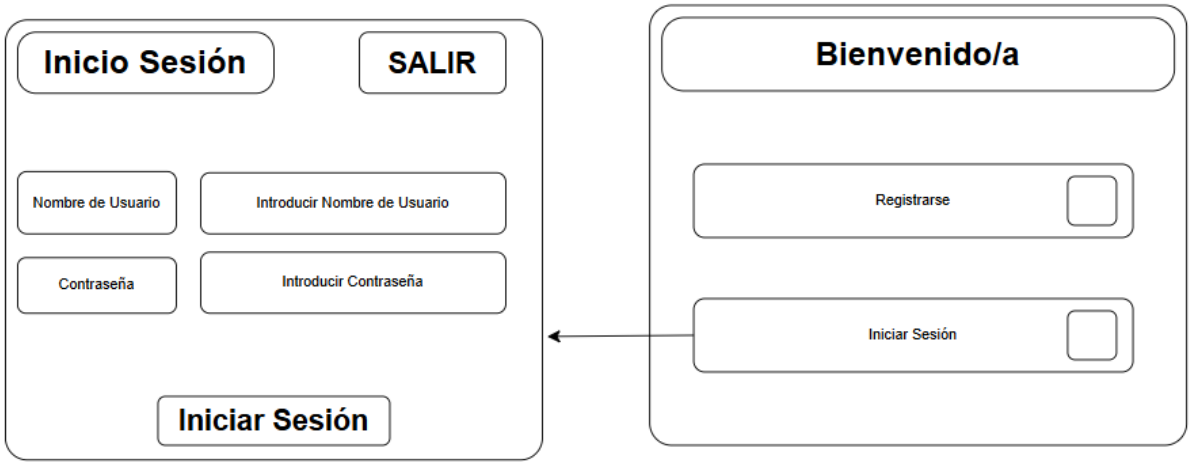

Figura 1.22: Interfaz Identificarse

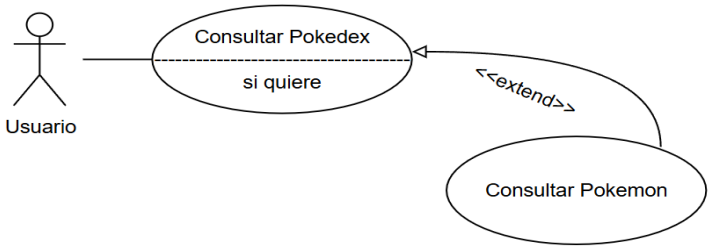

Figura 1.23: Diagrama Consultar Pokedex

1.12. Consultar Pokedex

- **Nombre**. Consultar Pokedex.
- **Descripción**. Consultar los datos de cualquier pokémon de nuestro sistema sea capturado o no.
- **Actores** . Usuario.
- **Precondiciones**.
  1. El usuario ha iniciado sesión en la aplicación.
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. El usuario selecciona la opción de consultar la pokedex.
  2. Se muestra la pestaña de la pokedex con la lista de pokemons.
- **Postcondiciones**.
  1. Éxito: Se muestra el menu de la pokedex.
  2. Error: Se muestra mensaje y se conserva el valor anterior.

Interfaz Gráfica

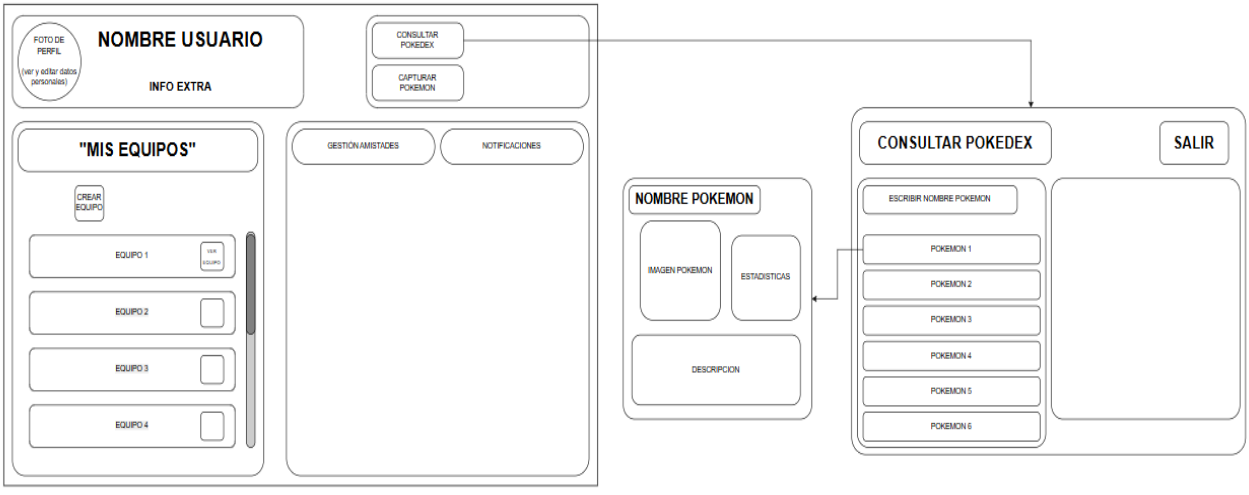

Figura 1.24: Interfaz Consultar Pokedex

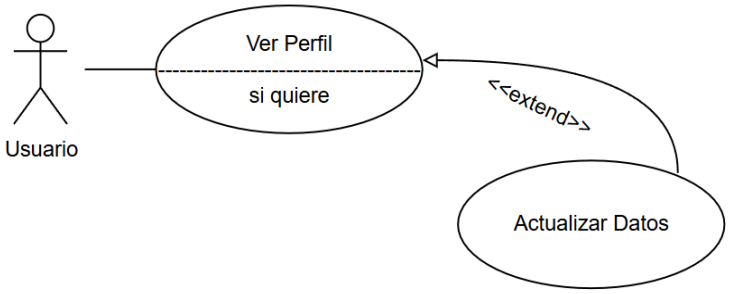

Figura 1.25: Diagrama Ver Perfil

1.13. Ver perfil

- **Nombre**. Ver perfil.
- **Descripción**. Menú que tiene cada perfil para poder consultar sus datos personales o editarlos si así lo desea
- **Actores** . Usuario.
- **Precondiciones**.
  1. El usuario está en la ventana "Ver perfil".
- **Requisitos no funcionales**. Ninguno
- **Flujo de eventos :**
  1. Usuario pincha en su foto de perfil
  2. Se muestra la ventana “Mi perfil” junto a los datos del usuario.
     a) Se muestra la opción de editar los datos personales.
- **Postcondiciones**.
  1. Éxito: El dato modificado se guarda y muestra actualizado.
  2. Cancelación: El dato mantiene su valor original.
  3. Error: Se muestra mensaje y se conserva el valor anterior.

Interfaz Gráfica

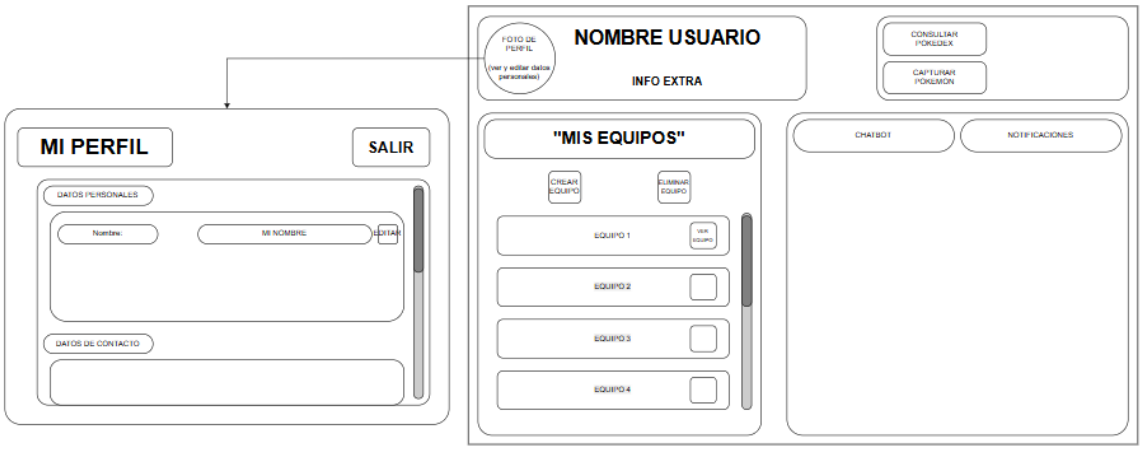

Figura 1.26: Interfaz Ver Perfil

Capítulo 2

Modelo de Dominio

2.1. Introducción

Este documento presenta la documentación del modelo de dominio para el Sistema Pokédex. El sistema permite a los usuarios gestionar sus propios Pokémon, organizarlos en equipos, consultar la Pokédex oficial, seguir a otros usuarios y mantener un registro de eventos.

2.2. Descripción del Modelo de Dominio

El modelo de dominio captura las entidades principales del sistema y sus relaciones.

2.2.1. Entidades Principales

Usuario

Representa a los usuarios registrados en el sistema.

- **nombreU**: Nombre de usuario único.
- **email**: Correo electrónico de contacto.
- **hash_contraseña**: Hash de seguridad de la contraseña.
- **nombreCompleto**: Nombre real o visual del usuario.
- **rol**: Rol del usuario (ej. administrador, entrenador).
- **fecha creación**: Fecha en la que se registró el usuario.

Equipo

Conjunto de Pokémon organizado por un usuario.

- **nombre_E**: Nombre asignado al equipo.
- **descripcion**: Breve descripción de la estrategia o temática.
- **esVisible**: Indicador de privacidad (público/privado).
- **ultima_actualizacion**: Timestamp de la última modificación.

Pokemon

Representa una instancia concreta de un Pokémon capturado por un usuario.

- **fechaCaptura**: Fecha en la que el usuario obtuvo el Pokémon.
- **movimientos**: Conjunto de movimientos aprendidos actualmente.
- **habilidad**: Habilidad activa específica de este Pokémon.
- **apodo**: Nombre personalizado dado por el entrenador.

Pokemon_Pokedex

Información de referencia (especie) inmutable del sistema.

- **nombreP**: Nombre de la especie (ej. Pikachu).
- **id**: Número identificador en la Pokédex.
- **peso**: Peso promedio de la especie.
- **altura**: Altura promedio de la especie.
- **sprite**: Referencia a la imagen visual del Pokémon.
- **tipo**: Tipo(s) elemental(es) (ej. Fuego, Agua).
- **listaHabilidad**: Lista de todas las habilidades posibles para la especie.
- **es_legendario**: Booleano, indica si es un Pokémon legendario.
- **es_mega**: Booleano, indica si es una mega evolución.
- **movimientos**: Lista de todos los movimientos que la especie puede aprender.

Changelog_Event

Entidad que registra el tipo de evento ocurrido en el sistema.

- **tipo_evento**: Categoría del evento realizado.

2.2.2. Clases de Asociación y Auxiliares

Index_Fecha

Clase de asociación que vincula un Pokémon con un Equipo.

- **Fecha**: Fecha exacta en la que el Pokémon fue añadido al equipo.

Entidades de Registro (Fecha y Descripcion)

Utilizadas en la relación de eventos para detallar cuándo y qué ocurrió.

- **Fecha** (Entidad Fecha): Marca temporal del evento.
- **Texto** (Entidad Descripcion): Detalle textual del evento.

2.3. Relaciones y Cardinalidades

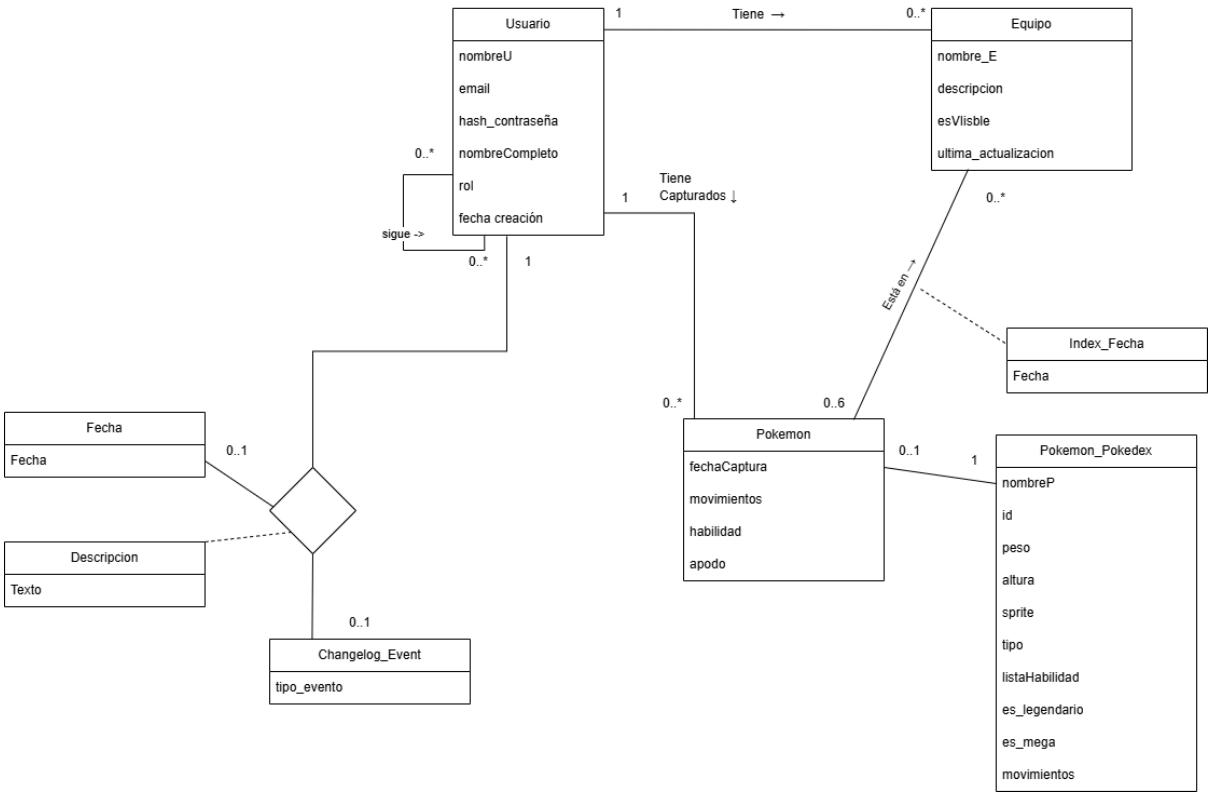

Figura 2.1: Diagrama del Modelo de Dominio Actualizado

Las relaciones definidas en el diagrama son:

- **Usuario - Equipo** (1 a 0..*): Un usuario es propietario de múltiples equipos. Un equipo pertenece a un solo usuario ("Tiene").
- **Usuario - Pokemon** (1 a 0..*): Un usuario posee múltiples instancias de Pokémon ("Tiene Capturados").
- **Usuario - Usuario** (0..* a 0..*): Relación reflexiva ("sigue"). Un usuario puede seguir a muchos usuarios y ser seguido por muchos.
- **Pokemon - Pokemon_Pokedex** (0..1 a 1): Cada instancia de Pokémon corresponde exactamente a una especie de la Pokédex.
- **Pokemon - Equipo** (0..6 a 0..*): Relación .Está en". Un equipo puede tener hasta 6 Pokémon. Un mismo Pokémon puede estar en múltiples equipos diferentes. Esta relación tiene una clase de asociación **Index_Fecha** para registrar cuándo se unió.
- **Relación de Eventos (Rombo N-aria)**: Existe una relación compleja que vincula a:
  - **Usuario** (1): El generador del evento.
  - **Changelog_Event** (0..1): El tipo de evento generado.
  - **Fecha** (0..1): Cuándo ocurrió.
  - **Descripcion** (0..1): Qué ocurrió (Texto).

2.4. Justificación de Diseño

2.4.1. Gestión de Equipos y Pokémon

A diferencia de modelos anteriores, la relación entre **Pokemon** y **Equipo** es directa pero acotada a una cardinalidad máxima de 6 en el lado del equipo, respetando las reglas clásicas de Pokémon. Se utiliza la entidad asociativa **Index_Fecha** para mantener un histórico de cuándo se incorporó cada miembro al equipo.

2.4.2. Sistema de Auditoría (Changelog)

Se ha modelado un sistema de registro mediante una relación n-aria (representada por el rombo). Esto permite que una acción de un **Usuario** genere una instancia única que agrupa el **tipo de evento**, la **descripción** textual y la **fecha**, asegurando la trazabilidad de las acciones importantes sin sobrecargar la entidad de usuario con atributos de log.

2.4.3. Separación de Instancia y Referencia

Se mantiene la distinción clara entre **Pokemon** (la instancia capturada con datos variables como apodo y fecha de captura) y **Pokemon_Pokedex** (la ficha técnica inmutable de la especie), optimizando el almacenamiento de datos repetitivos.

Capítulo 3

Plan de Pruebas

3.1. Gestión de Usuarios (1)

3.1.1. Registro e Inicio de Sesión

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Registrarse', CU 'Identificarse')
Introducción: Este plan detalla las pruebas diseñadas para validar que el sistema gestiona correctamente la creación de nuevas cuentas y el acceso de usuarios, cumpliendo con la regla de negocio específica de que las cuentas nuevas requieren aprobación de un administrador antes de ser funcionales.

Cuadro 3.1: Casos de Prueba: Registro e Inicio de Sesión

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-REG-01	Registro	**(Caso Positivo)**. Introducir datos válidos y nuevos en todos los campos (Nombre, Apellidos, Usuario, Email, Contraseña coincidente) y pulsar Crear Cuenta. "	El sistema muestra un mensaje de éxito indicando que la cuenta está pendiente de aprobación y redirige a la pantalla de inicio."
CP-REG-02	Registro	**(Caso Negativo - Campos Vacíos)**. Intentar registrarse dejando uno o varios campos obligatorios vacíos (ej: no introducir Email).	El sistema no crea la cuenta y muestra un mensaje de error indicando los campos requeridos.
CP-REG-03	Registro	**(Caso Negativo - Contraseñas)**. Introducir una contraseña válida en el primer campo, pero una diferente en el campo Repetir Contraseña. "	El sistema muestra un error indicando que las contraseñas no coinciden y no permite el registro.

Continúa en la siguiente página

Cuadro 3.1 – Continuación

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-REG-04	Registro	**(Caso Frontera - Formato Email)**. Introducir un formato de email inválido (ej: usuario.com sin arroba, o @dominio.com sin usuario local). "	El sistema detecta el formato erróneo, muestra error y no permite el registro."
CP-REG-05	Registro	**(Caso Negativo - Duplicidad)**. Ïntentar registrarse con un Nombre de Usuario o Email que ya existe en el sistema."	El sistema muestra un mensaje de error indicando que el usuario/email ya está en uso.
CP-REG-06	Registro	**(Caso Frontera - Límites de Caracteres)**. Introducir nombres de usuario extremadamente largos (ej: +255 caracteres) o vacíos (espacios en blanco).	El sistema debe manejarlo adecuadamente (cortando el input o mostrando error de longitud máxima/mínima).
CP-REG-07	Registro	**(Navegación)**. Pulsar el botón SALIR en el formulario de registro sin completar datos."	El sistema regresa a la pantalla de bienvenida sin crear ninguna cuenta ni mostrar errores.
CP-LOG-01	Inicio Sesión	**(Caso Positivo)**. Introducir usuario y contraseña de una cuenta que ya ha sido aprobada por el administrador.	Acceso correcto. El sistema redirige al Menú Principal del usuario.
CP-LOG-02	Inicio Sesión	**(Caso Negativo - Pendiente)**. Introducir usuario y contraseña correctos de una cuenta recién creada (NO aprobada por admin). **CRÍTICO:**	El sistema deniega el acceso y muestra un mensaje específico: Cuenta pendiente de aprobación (o similar).
CP-LOG-03	Inicio Sesión	**(Caso Negativo - Contraseña)**. Introducir un usuario existente, pero con una contraseña errónea.	El sistema muestra mensaje de error genérico (Credenciales incorrectas) y permanece en la pantalla de login.
CP-LOG-04	Inicio Sesión	**(Caso Negativo - Usuario inexistente)**. Introducir un nombre de usuario que no está registrado en la base de datos.	El sistema muestra mensaje de error y no permite el acceso.
CP-LOG-05	Inicio Sesión	**(Caso Frontera - Inyección/Caracteres Raros)**. Intentar iniciar sesión con caracteres especiales en el usuario (ej: ' OR 1=1 –) para probar seguridad básica.	El sistema maneja los caracteres como texto normal, no se rompe (crash) y deniega el acceso si no corresponde a un usuario real.

Continúa en la siguiente página

Cuadro 3.1 – Continuación

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-LOG-06	Inicio Sesión (UI)	**(Inicio Sesión (UI))**. Pulsar el botón SALIR en la pantalla de Inicio de Sesión (Fig. 1.22).	El sistema regresa a la pantalla de Bienvenido/a.
CP-NAV-01	Navegación Principal	**(Navegación Principal)**. Desde la pantalla principal (Bienvenido/a), verificar que los botones Registrarse e Iniciar Sesión llevan a sus pantallas correctas.	Navegación correcta según el diagrama de flujo de la Fig. 1.8 y 1.22.

3.1.2. Actualizar Datos Personales

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Ver Perfil', CU 'Actualizar Datos')
Introducción: Este plan detalla las pruebas diseñadas para validar que un usuario autenticado puede visualizar y modificar correctamente su información personal desde su perfil.

Cuadro 3.2: Casos de Prueba: Actualizar Datos Personales

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-PRF-01	Ver Perfil	**(Caso Positivo)**. Desde el menú principal, hacer clic en la foto/icono del usuario."	El sistema navega a la pantalla MI PERFIL mostrando los datos actuales correctos del usuario.
CP-ACT-01	Actualizar Datos	**(Caso Positivo - Cambio Simple)**. En la pantalla MI PERFIL, modificar un campo permitido (ej: Nombre o Datos de Contacto) con nueva información válida y pulsar Guardar/Confirmar.	El sistema muestra un mensaje de éxito, los datos se actualizan en la base de datos y la interfaz refleja los nuevos valores inmediatamente.
CP-ACT-02	Actualizar Datos	**(Caso Negativo - Campo Vacío)**. Ïntentar guardar el perfil dejando un campo obligatorio vacío (ej: borrar todo el Nombre).	El sistema muestra un mensaje de error indicando que el campo no puede estar vacío y no guarda los cambios.
CP-ACT-03	Actualizar Datos	**(Caso Negativo - Formato Inválido)**. Si se permite editar el email, introducir un formato inválido (ej: correo.sin. arroba).	El sistema detecta el formato incorrecto, muestra un error y mantiene el valor anterior."
CP-ACT-04	Actualizar Datos	**(Navegación - Cancelar)**. Modificar varios campos, pero pulsar el botón SALIR o CANCELAR en lugar de guardar.	El sistema regresa a la pantalla anterior y los datos del perfil NO se han modificado (verificar volviendo a entrar al perfil).
CP-ACT-05	Actualizar Datos	**(Caso Frontera - Límites)**. Introducir la máxima cantidad de caracteres permitidos en un campo de texto (ej: una descripción muy larga en Info Extra si existe)."	El sistema permite guardar hasta el límite definido sin cortar el texto abruptamente ni dar error de base de datos.
CP-ACT-06	Actualizar Datos	**(Seguridad - Inyección)**. Intentar guardar código HTML o SQL en los campos de texto (ej: `<b>Negrita</b>` o `' OR 1=1`).	El sistema guarda el texto literalmente, pero no ejecuta el código (no se ve negrita en el perfil, ni ocurre error de SQL).

Cuadro 3.2 – Continuación

ID Prueba	**Funcionalidad**	**Descripción de la Prueba (Input/Acción)**	**Resultado Esperado (Output)**
CP-PRF-02	Ver Perfil (UI)	**(Ver Perfil (UI))**. Pulsar el botón SALIR en la pantalla MI PERFIL.	El sistema regresa al menú principal (donde se ven MIS EQUIPOS, etc.).

3.1.3. Administración

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Eliminar Cuentas', CU 'Ver Datos', CU 'Modificar Datos') y Diagrama de Jerarquía (CU 'Aprobar Cuenta')
Introducción: Este plan valida las funcionalidades exclusivas del rol de Administrador para gestionar el ciclo de vida de las cuentas de usuario: aprobación de nuevos registros, modificación de datos existentes y eliminación de usuarios.

Cuadro 3.3: Casos de Prueba: Administración

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-ADM-01	Aprobar Cuenta	**(Caso Positivo)**. Iniciar sesión como Administrador, ir a la lista de usuarios "Pendientes", seleccionar un usuario recién registrado y pulsar Aprobar.	El usuario desaparece de la lista de Pendientes, aparece en la lista principal de Usuarios activos y ahora puede iniciar sesión (verificar con CP-LOG-01).
CP-ADM-02	Aprobar Cuenta	**(Caso Negativo - Rechazar)**. En la lista de "Pendientes", seleccionar un usuario y pulsar Rechazar o Eliminar (si existe la opción directa).	La cuenta se elimina permanentemente y el usuario no puede iniciar sesión.
CP-ADM-03	Eliminar Cuenta	**(Caso Positivo)**. En la lista principal de Usuarios, seleccionar un usuario activo y pulsar el botón ELIMINAR. Confirmar la acción si pide confirmación.	El usuario desaparece de la lista. Intentar iniciar sesión con esa cuenta posteriormente debe fallar (CP-LOG-04).
CP-ADM-04	Eliminar Cuenta	**(Caso Negativo - Cancelar)**. Pulsar ELIMINAR sobre un usuario, pero seleccionar Cancelar en el mensaje de confirmación.	El usuario permanece activo en la lista y puede seguir iniciando sesión.
CP-ADM-05	Eliminar Cuenta	**(Caso Frontera - Autoeliminación)**. Intentar que el Administrador se elimine a sí mismo de la lista de usuarios.	El sistema debe impedir esta acción (botón deshabilitado o mensaje de error) para evitar quedarse sin administradores.
CP-ADM-06	Modificar Datos	**(Caso Positivo)**. Seleccionar VER USUARIO de una cuenta activa, cambiar un dato (ej: su email) y guardar.	Los cambios se guardan correctamente. El usuario afectado verá el nuevo dato en su perfil.
CP-ADM-07	Modificar Datos	**(Caso Negativo - Datos Inválidos)**. Intentar cambiar el nombre de usuario de una cuenta por uno que YA existe en otra cuenta.	El sistema muestra un error de duplicidad y no permite guardar el cambio.

3.1.4. Gestión de Amigos

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Gestión Amistades') e Interfaz Gráfica (Pantalla 'AMISTADES')
Introducción: Este plan cubre el ciclo completo de interacción entre usuarios para establecer relaciones de amistad: envío de solicitudes, aceptación/rechazo de las mismas y visualización de la lista de amigos.

Cuadro 3.4: Casos de Prueba: Gestión de Amigos

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-AMI-01	Enviar Solicitud	**(Caso Positivo)**. En la pantalla Enviar Solicitud Amigo, introducir el nombre exacto de un usuario existente que NO es amigo aún y pulsar Enviar.	El sistema muestra un mensaje de éxito (Solicitud enviada). El usuario receptor debería ver una nueva notificación (verificable con otra cuenta).
CP-AMI-02	Enviar Solicitud	**(Caso Negativo - Usuario Inexistente)**. Intentar enviar solicitud a un nombre de usuario que no existe en la base de datos.	El sistema muestra un mensaje de error indicando que el usuario no fue encontrado.
CP-AMI-03	Enviar Solicitud	**(Caso Negativo - Ya Amigos/-Solicitado)**. Intentar enviar solicitud a un usuario que YA es amigo o al que YA se le ha enviado una solicitud pendiente.	El sistema muestra un mensaje indicando que ya existe una relación con ese usuario.
CP-AMI-04	Enviar Solicitud	**(Caso Frontera - Auto solicitud)**. Intentar enviar una solicitud de amistad a uno mismo (introduciendo el propio nombre de usuario).	El sistema debe impedir esta acción y mostrar un mensaje de error adecuado.
CP-AMI-05	Aceptar Solicitud	**(Caso Positivo)**. Entrar con una cuenta que tenga solicitudes recibidas. En Solicitudes Pendientes, pulsar Aceptar en una de ellas.	La solicitud desaparece de Pendientes y el usuario solicitante aparece ahora en la lista de Consultar Amistades.
CP-AMI-06	Rechazar Solicitud	**(Caso Positivo)**. En Solicitudes Pendientes, pulsar Rechazar (o icono de cruz/eliminar) en una solicitud.	La solicitud desaparece de la lista y NO se crea la relación de amistad.
CP-AMI-07	Consultar Amigos	**(Verificación)**. Acceder a Consultar Amistades después de aceptar una solicitud (CP-AMI-05).	La lista debe mostrar correctamente el nombre del nuevo amigo añadido.

Cuadro 3.4 – Continuación

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-AMI-08	Eliminar Amigo	**(Si aplica)**. Desde la lista de Consultar Amistades, seleccionar un amigo y pulsar Eliminar Amigo (si existe esa función en la interfaz final)."	El usuario desaparece de la lista de amigos. Ambos usuarios dejan de ser amigos.
CP-AMI-09	Interfaz Amigos	**(Interfaz Amigos)**. Verificar que al entrar en GESTIÓN AMISTADES, se ven las tres secciones principales (Fig. 1.14) y se puede navegar entre ellas.	La navegación interna del módulo de amigos funciona correctamente.

3.2. Gestión de Equipos Pokémon (2)

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Crear Equipo', CU 'Consultar Equipo', CU 'Editar Equipos') e Interfaces Gráficas (Fig. 1.4, 1.6).
Introducción: Este plan valida el ciclo completo de vida de los equipos Pokémon: creación (respetando el límite de 6 miembros), consulta, edición y eliminación, asegurando que la interfaz responde correctamente a las acciones del usuario.

Cuadro 3.5: Casos de Prueba: Gestión de Equipos Pokémon

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-EQP-01	Crear Equipo	**(Caso Positivo - Estándar)**. Desde MIS EQUIPOS, pulsar CREAR EQUIPO. Introducir un nombre válido y seleccionar entre 1 y 6 Pokémon. Pulsar CONFIRMAR CAMBIOS.	El nuevo equipo aparece en la lista MIS EQUIPOS con el nombre correcto y el número de Pokémon seleccionados.
CP-EQP-02	Crear Equipo	**(Caso Frontera - Límite Máximo)**. Intentar añadir un 7º Pokémon a un equipo que ya tiene 6 seleccionados durante la creación o edición.	El sistema NO debe permitir seleccionar el 7º Pokémon (el botón de añadir se deshabilita o muestra un mensaje de Equipo lleno).
CP-EQP-03	Crear Equipo	**(Caso Negativo - Nombre Vacío)**. Intentar crear un equipo sin introducir nada en el campo NOMBRE EQUIPO POKEMON.	El sistema muestra un mensaje de error indicando que el nombre es obligatorio y no crea el equipo.
CP-EQP-04	Crear Equipo	**(Caso Negativo - Equipo Vacío)**. Introducir un nombre de equipo, pero NO seleccionar ningún Pokémon (0 miembros) y pulsar CONFIRMAR.	Dependiendo de vuestra regla de negocio: o muestra error Debe elegir al menos 1 Pokémon, o crea un equipo vacío válido. (Idealmente debería exigir al menos 1).
CP-EQP-05	Crear Equipo	**(Caso Negativo - Nombre Duplicado)**. Intentar crear un segundo equipo con exactamente el mismo nombre que uno ya existente del mismo usuario.	El sistema debería mostrar un aviso sugiriendo usar un nombre diferente para evitar confusión, aunque podría permitirlo si no es restricción técnica. (Recomendable: No permitir duplicados).

Cuadro 3.5 – Continuación

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-EQP-06	Consultar Equipo	**(Caso Positivo)**. Seleccionar un equipo existente de la lista MIS EQUIPOS.	Se abre la vista de detalle mostrando el nombre del equipo y la lista correcta de los Pokémon que lo componen.
CP-EQP-07	Editar Equipo	**(Caso Positivo - Añadir/Quitar)**. En un equipo existente, entrar a editar, eliminar un Pokémon y añadir otro diferente. Guardar cambios.	El equipo se actualiza correctamente. Al volver a consultarlo, muestra la nueva composición.
CP-EQP-08	Editar Equipo	**(Caso Positivo - Cambiar Nombre)**. Editar solo el nombre de un equipo existente y guardar.	El nombre se actualiza en la lista principal de MIS EQUIPOS.
CP-EQP-09	Eliminar Equipo	**(Caso Positivo)**. Seleccionar un equipo y pulsar el botón de eliminar (papelera o similar). Confirmar si hay mensaje.	El equipo desaparece inmediatamente de la lista MIS EQUIPOS.
CP-EQP-10	Navegación	**(Cancelar Creación/Edición)**. Estar en medio de crear o editar un equipo (habiendo seleccionado algunos Pokémon) y pulsar SALIR o CANCELAR.	El sistema regresa a la pantalla anterior sin guardar ningún cambio (el equipo nuevo no se crea, o el editado mantiene su estado previo).

3.3. ChangeLog (3)

3.3.1. Generación de Notificaciones

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Ver Notificaciones', CU 'Filtrar Notificaciones') e Interfaz Gráfica (Fig. 1.20).
Introducción: Este plan valida que las acciones clave realizadas por un usuario (crear/editar equipos, capturar Pokémon) generan automáticamente las notificaciones correspondientes en el ChangeLog de todos sus seguidores.

Cuadro 3.6: Casos de Prueba: Generación de Notificaciones

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-LOG-GEN-01	Notificación Equipo (Creación)	**(Precondición: Usuario A sigue a Usuario B)**. Usuario B crea un nuevo equipo Pokémon y guarda los cambios. Usuario A accede a su ChangeLog.	Usuario A ve una nueva notificación: Usuario B ha creado el equipo [NombreEquipo]."
CP-LOG-GEN-02	Notificación Equipo (Edición)	**(Precondición: Usuario A sigue a Usuario B)**. Usuario B edita un equipo existente (cambia nombre o miembros). Usuario A accede a su ChangeLog.	Usuario A ve una nueva notificación: Usuario B ha actualizado su equipo [NombreEquipo]."
CP-LOG-GEN-03	Notificación Captura	**(Precondición: Usuario A sigue a Usuario B)**. Usuario B usa la función Capturar Pokémon y obtiene uno nuevo. Usuario A accede a su ChangeLog.	Usuario A ve una nueva notificación: Usuario B ha capturado a [NombrePokemon]."
CP-LOG-GEN-04	Alcance Notificación	**(Precondición: Usuario C NO sigue a Usuario B)**. Usuario B realiza una acción (equipo/captura) que genera notificación. Usuario C accede a su ChangeLog.	Usuario C NO ve ninguna notificación relacionada con las acciones recientes de Usuario B.
CP-LOG-GEN-05	Notificación Propia	**(Notificación Propia)**. Usuario B realiza una acción (equipo/captura) y accede a SU PROPIO ChangeLog.	(Dependiendo de la regla de negocio): Usuario B NO debería ver sus propias notificaciones, solo las de la gente a la que sigue.

3.3.2. Filtrado y Visualización

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Ver Notificaciones', CU 'Filtrar Notificaciones') e Interfaz Gráfica (Fig. 1.20).
Introducción: Este plan valida las herramientas disponibles para que el usuario gestione su visualización del ChangeLog, específicamente el filtrado por usuario y la navegación en la interfaz.

Cuadro 3.7: Casos de Prueba: Filtrado y Visualización

ID Prueba	**Funcionalidad**	**Descripción de la Prueba (Input/Acción)**	**Resultado Esperado (Output)**
CP-LOG-VIS-01	Ver ChangeLog	**(Caso Positivo)**. Desde el menú principal, pulsar el botón NOTIFICACIONES.	Se abre la pantalla de historial de notificaciones mostrando las más recientes primero (orden cronológico inverso).
CP-LOG-FIL-01	Filtrar por Usuario	**(Caso Positivo)**. En la pantalla de notificaciones, pulsar FILTRAR NOTIFICACIONES y seleccionar un amigo específico de la lista.	La lista se actualiza y SOLO muestra las notificaciones generadas por ese amigo seleccionado.
CP-LOG-FIL-02	Filtrar (Sin datos)	**(Caso Vacío)**. Seleccionar un filtro de usuario que NO ha generado ninguna notificación reciente.	La lista aparece vacía o muestra un mensaje informativo (No hay actividad reciente de este usuario).
CP-LOG-FIL-03	Quitar Filtros	**(Caso Positivo)**. Después de aplicar un filtro (CP-LOG-FIL-01), usar la opción para limpiar/quitar filtros (ej: botón Ver Todos o deseleccionar).	La lista vuelve a mostrar todas las notificaciones de todos los seguidos.
CP-LOG-NAV-01	Navegación (Salir)	**(Navegación (Salir))**. Pulsar el botón SALIR en la pantalla de HISTORIAL NOTIFICACIONES (Fig. 1.20).	El sistema regresa correctamente al menú principal.

3.4. Lista Completa de Pokémon (4)

3.4.1. Mostrar Lista

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Consultar Pokedex') e Interfaz Gráfica (Fig. 1.24).
Introducción: Este plan valida que el sistema es capaz de conectar con la PokeAPI y renderizar correctamente la lista completa de Pokémon en la interfaz de usuario, gestionando la carga de datos y posibles errores de conexión.

Cuadro 3.8: Casos de Prueba: Mostrar Lista

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-API-LIST-01	Carga Inicial API	**(Caso Positivo)**. Desde el menú principal, acceder a CONSULTAR POKEDEX.	La lista se carga mostrando los primeros Pokémon en orden numérico estándar (ej: `#1` Bulbasaur, `#2` Ivysaur...). Se ven sus nombres e imágenes correctamente.
CP-API-LIST-02	Paginación/Scroll	**(Caso Positivo)**. Hacer scroll hacia el final de la lista cargada inicialmente (o pulsar botón Cargar más si vuestra UI lo usa).	El sistema realiza una nueva petición a la API y añade el siguiente lote de Pokémon a la lista sin reemplazar los anteriores.
CP-API-LIST-03	Manejo de Errores	**(Caso Negativo - Sin Conexión)**. Desconectar el equipo de internet o simular fallo de la API, e intentar entrar a CONSULTAR POKEDEX.	La aplicación NO se cierra inesperadamente (crash). Muestra un mensaje amigable al usuario (ej: Error de conexión con la Pokedex) y permite volver al menú.
CP-API-LIST-04	Rendimiento	**(Prueba de Estrés Básica)**. Entrar y salir repetidamente de la sección CONSULTAR POKEDEX (ej: 5 veces seguidas rápidamente).	La aplicación gestiona bien las múltiples peticiones sin bloquearse ni mostrar datos erróneos de cargas anteriores.
CP-API-UI-01	Navegación (UI)	**(Navegación (UI))**. Pulsar el botón SALIR en la pantalla de la lista completa (Fig. 1.24).	El sistema regresa correctamente al menú principal.

3.4.2. Filtros de Búsqueda

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso Extendidos (CU 'Consultar Pokedex' - menciona búsqueda) e Interfaz Gráfica (Fig. 1.24
Introducción: Este plan valida que los mecanismos de filtrado permiten acotar la lista completa de Pokémon correctamente, asegurando que los resultados mostrados coinciden exactamente con los criterios seleccionados por el usuario.

Cuadro 3.9: Casos de Prueba: Filtros de Búsqueda

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-FIL-01	Filtro por Nombre	**(Caso Positivo - Parcial)**. Ïntroducir una cadena parcial en el campo de búsqueda (ej: cha). "	La lista se actualiza mostrando solo Pokémon que contienen esa cadena (Charmander, Charizard, Chikorita, Chimchar...).
CP-FIL-02	Filtro por Nombre	**(Caso Negativo - Sin resultados)**. Introducir una cadena que no corresponde a ningún Pokémon (ej: xyz123).	La lista se vacía y muestra un mensaje informativo (No se encontraron resultados).
CP-FIL-03	Filtro por Tipo	**(Caso Positivo)**. Seleccionar un tipo concreto en el filtro correspondiente (ej: Tipo Fuego).	La lista muestra únicamente Pokémon que tienen Fuego como tipo primario o secundario."
CP-FIL-04	Filtro por Habilidad	**(Caso Positivo)**. Seleccionar una habilidad específica (ej: Intimidación).	La lista muestra solo los Pokémon que pueden tener esa habilidad.
CP-FIL-05	Filtros Combinados	**(Caso Positivo - Intersección)**. Aplicar dos filtros compatibles simultáneamente (ej: Tipo Eléctrico Y Nombre contiene Pika).	La lista muestra solo los resultados que cumplen AMBAS condiciones (Pikachu, Raichu...).
CP-FIL-06	Filtros Combinados	**(Caso Negativo - Incompatibles)**. Aplicar filtros que no pueden darse juntos (ej: Tipo Fuego Y Habilidad Torrente - exclusiva de agua).	La lista muestra 0 resultados y el mensaje de No encontrado , sin producir errores en la aplicación.
CP-FIL-07	Limpiar Filtros	**(Caso Positivo)**. Después de aplicar uno o varios filtros, usar la opción de Limpiar filtros (o borrar manualmente el texto/deseleccionar opciones).	La lista vuelve a mostrar todos los Pokémon disponibles (estado inicial).

Cuadro 3.9 – Continuación

ID Prueba	**Funcionalidad**	**Descripción de la Prueba (Input/Acción)**	**Resultado Esperado (Output)**
CP-FIL-08	Rendimiento Filtros	**(Respuesta UI)**. Escribir rápidamente en el buscador por nombre o cambiar filtros de tipo velozmente.	La interfaz responde de forma fluida, actualizando la lista sin congelarse notablemente entre pulsaciones.

3.4.3. Ver Detalles de Pokémon

Versión del Plan: 1.0
Tipo de Pruebas: Funcionales (Caja Negra)
Documentación Base: Casos de Uso ExtendIDOS (CU 'Consultar Pokémon', extendido desde 'Consultar Pokedex' y 'Consultar Equipo') e Interfaz Gráfica (Fig. 1.2 y 1.24).
Introducción: Este plan valida que, al seleccionar un Pokémon específico, el sistema recupera y muestra correctamente toda su información detallada (imágenes, estadísticas, descripciones), independientemente de desde qué parte de la aplicación se acceda.

Cuadro 3.10: Casos de Prueba: Ver Detalles de Pokémon

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-DET-01	Acceso desde Pokedex	**(Caso Positivo - Flujo Principal)**..En la lista completa CONSULTAR POKEDEX, hacer clic en un Pokémon (ej: Charizard).	Se abre la vista de detalles mostrando correctamente: Nombre (Charizard), Imagen de alta resolución, Estadísticas base y Descripción.
CP-DET-02	Acceso desde Equipo	**(Caso Positivo - Integración)**. Desde MIS EQUIPOS, entrar en un equipo y hacer clic en uno de los Pokémon miembros.	Se debe reutilizar la misma vista de detalles y mostrar la información correcta del Pokémon seleccionado desde el equipo.
CP-DET-03	Integridad de Datos	**(Verificación de Contenido)**. Hay que confirmar que los datos mostrados en la vista de detalles corresponden exactamente al Pokémon clicado.	No debe haber "mezcla"de datos (ej: ver la imagen de Pikachu con el nombre de Bulbasaur) si se navega rápido.
CP-DET-04	Datos Incompletos (API)	**(Caso Frontera - API)**. Acceder a un Pokémon de generaciones muy recientes o variantes raras que podrían tener datos incompletos en la API (ej: sin descripción en español).	La interfaz maneja la falta de datos elegantemente (mostrando "Descripción no disponible.º texto en inglés por defecto) sin romperse o mostrar huecos vacíos feos.
CP-DET-05	Navegación (Volver)	**(Flujo de UI)**. Estando en la vista de detalles (tras acceder desde la Pokedex filtrada, ej: filtro Agua), pulsar SALIR o VOLVER.	El sistema regresa a la lista de Pokedex MANTENIENDO los filtros que estaban activos (solo ver Pokémon de agua), no resetea la lista completa.

Cuadro 3.10 – Continuación

ID Prueba	Funcionalidad	Descripción de la Prueba (Input/Acción)	Resultado Esperado (Output)
CP-DET-06	Error de Carga	**(Caso Negativo)**. Simular fallo de red justo al hacer clic en un Pokémon para ver detalles.	Se muestra un mensaje de error específico en la modal/pantalla de detalles (ej: No se pudo cargar la información del Pokémon) permitiendo reintentar o salir.

Implementación del plan de pruebas

Se han ejecutado varios de los casos de prueba definidos en el plan, obteniendo resultados positivos en la totalidad de ellos sin incidencias. Con esto, finalizamos el plan de pruebas garantizando la estabilidad de las funcionalidades implementadas.

Historial de Prompts Utilizados

1º prompt utilizado

Actúa como un QA Tester. Necesito un plan de pruebas de caja negra para el proyecto pokedex. Basándote en la teoría estándar de pruebas de software, genera una tabla con casos de prueba para las diferentes funcionalidades. Vamos a empezar por Registro e inicio de sesión (Que pertenece al punto 1). La tabla debe tener las columnas: ID, Funcionalidad, Descripción de la prueba (Input) y Resultado Esperado. Asegúrate de incluir casos positivos, negativos (búsquedas que fallan) y casos frontera (inputs vacíos o raros).

(compartimos el documento de casos de uso) genero la tabla de registro/inicio sesión.

2º prompt utilizado

si, ahora dentro del punto 2 este apartado: Los usuarios podrán crear equipos formados por un máximo de 6 Pokémon. Fíjate además en los casos de uso y de la interfaz gráfica para comprobar donde podría fallar el programa y haz las pruebas en base a eso también.

A partir de aquí se empezó a fijar en la interfaz gráfica también.

Capítulo 4

Diagrama de Clases

4.1. Diseño Detallado de Clases

En este capítulo se detalla la arquitectura de software propuesta para el *Sistema Pokédex*, cubriendo las funcionalidades de gestión de usuarios, creación de equipos y sistema de notificaciones (ChangeLog). El diseño sigue una estrategia modular para desacoplar la lógica de negocio de la interfaz de usuario y la persistencia de datos.

A continuación, se describen los tres subsistemas principales que componen la solución para los objetivos 1 a 4 del proyecto.

4.1.1. Subsistema de Gestión de Usuarios y Autenticación

La arquitectura para la gestión de usuarios (Funcionalidad 1) se ha diseñado siguiendo el patrón **Modelo-Vista-Controlador (MVC)** para separar claramente la lógica de control de la representación visual. Como se puede observar en la Figura 4.1, el sistema se estructura en capas bien definidas.

Figura 4.1: Diagrama de clases del módulo de Gestión de Usuarios y Roles (MVC).

Descripción de Componentes

Modelo (Entidades)

La clase `Usuario` es el núcleo del modelo, almacenando credenciales protegidas (`password_hash`), información de perfil y el rol del sistema. La gestión de "seguidores"se modela mediante la entidad `Amistades`, que rompe la relación muchos-a-muchos entre usuarios, registrando el `sender_id`, `receiver_id` y el estado temporal de la solicitud (*pending*, *accepted*).

Controladores (Lógica de Negocio)

Se han especializado tres controladores para evitar çlases dios":

- `ControladorAuth`: Encapsula exclusivamente el ciclo de vida de la sesión (login, registro, logout) y la validación de credenciales.
- `ControladorAdmin`: Contiene la lógica sensible accesible solo por administradores, como `aprobarCuenta` o `eliminarCuenta`.
- `ControladorSocial`: Gestiona la lógica de la red social, permitiendo `enviarSolicitud` o `rechazarSolicitud` de amistad.

Vistas y Enrutamiento

Las clases `RutasAdmin`, `RutasAuth` y `RutasPerfil` actúan como interfaz HTTP, recibiendo las peticiones del navegador y delegando la acción al controlador pertinente antes de renderizar las plantillas HTML finales.

4.1.2. Subsistema de ChangeLog y Eventos

Para la implementación del historial de cambios y notificaciones (Funcionalidad 3), se ha optado por una arquitectura centralizada basada en el patrón **Singleton** para los gestores de recursos, garantizando una única instancia de acceso a la base de datos y eventos. La estructura se detalla en la Figura 4.2.

Figura 4.2: Arquitectura del sistema de notificaciones y gestión de eventos.

La clase `SISTEMA_POKEDEX` actúa como una **Fachada** (Facade Pattern), simplificando la complejidad del subsistema para la interfaz de usuario (`IU_CHANGELOG`).

@lX@

Componente	Responsabilidad Técnica
`GESTOR_EVENTOS`	*Singleton*. Centraliza la lógica de publicación y suscripción de noticias. Valida los tipos de eventos mediante `validarTipoEvento` antes de persistirlos.
`GESTOR_USUARIOS`	*Singleton*. Provee servicios transversales, específicamente la recuperación de la lista de seguidos para filtrar el *feed* de noticias personal.
`GESTOR_BD`	*Singleton*. Capa de abstracción de datos (DAL) que encapsula las sentencias SQL puras y devuelve objetos `RESULTADO_SQL`, desacoplando la lógica del motor de base de datos.
`EVENTO`	Objeto de Transferencia de Datos (DTO) que serializa la información de una acción (tipo, descripción, fecha) a formato JSON para su consumo en el frontend.

Cuadro 4.1: Definición de clases del subsistema ChangeLog

4.1.3. Subsistema de Equipos y Datos Pokémon

Este módulo (Figura 4.3) soporta las funcionalidades de creación de equipos (Funcionalidad 2) y búsqueda de información (Funcionalidad 4). El diseño realiza una distinción arquitectónica crítica entre la definición biológica de la especie y la instancia concreta poseída por un jugador.

Figura 4.3: Diagrama de clases para la gestión de Equipos y estructura de Datos Pokémon.

Detalle de las Entidades

`Equipo` Representa la unidad organizativa principal del usuario. Cada instancia vincula un `UserId` con una colección dinámica de objetos `Pokemon`. Incluye métodos transaccionales como `editarEquipo()` y `eliminarEquipo()` para asegurar la consistencia de los datos en la base de datos.

`Pokemon` **(Instancia Dinámica)** Modela un Pokémon específico capturado por un entrenador. A diferencia de la especie genérica, esta clase almacena el estado mutable y único de la criatura:

- **Personalización:** Atributos como `apodo` permiten diferenciar múltiples capturas de la misma especie.
- **Combate:** Mantiene la `listaMovimientos` seleccionada (limitada a 4) y la `habilidad` activa concreta.
- **Historial:** Registra metadatos como la `fechaCaptura`.

`Especie` (Datos Estáticos API) Clase de solo lectura que mapea la información global obtenida de la *PokeAPI*. Sirve como catálogo de referencia para las búsquedas, almacenando:

- Estadísticas base (`peso`, `altura`) y tipos elementales.
- Recursos multimedia (`sprite`) para la visualización.
- Flags lógicos (`es_legendario`, `es_mega`) utilizados en los filtros de búsqueda avanzada.

Capítulo 5

Esquema relacional de la BBDD

En base al Modelo de Dominio hemos creado el siguiente esquema. Hay que tener en cuenta que PK es primary key y FK foreign key de cada tabla creada por entidad.

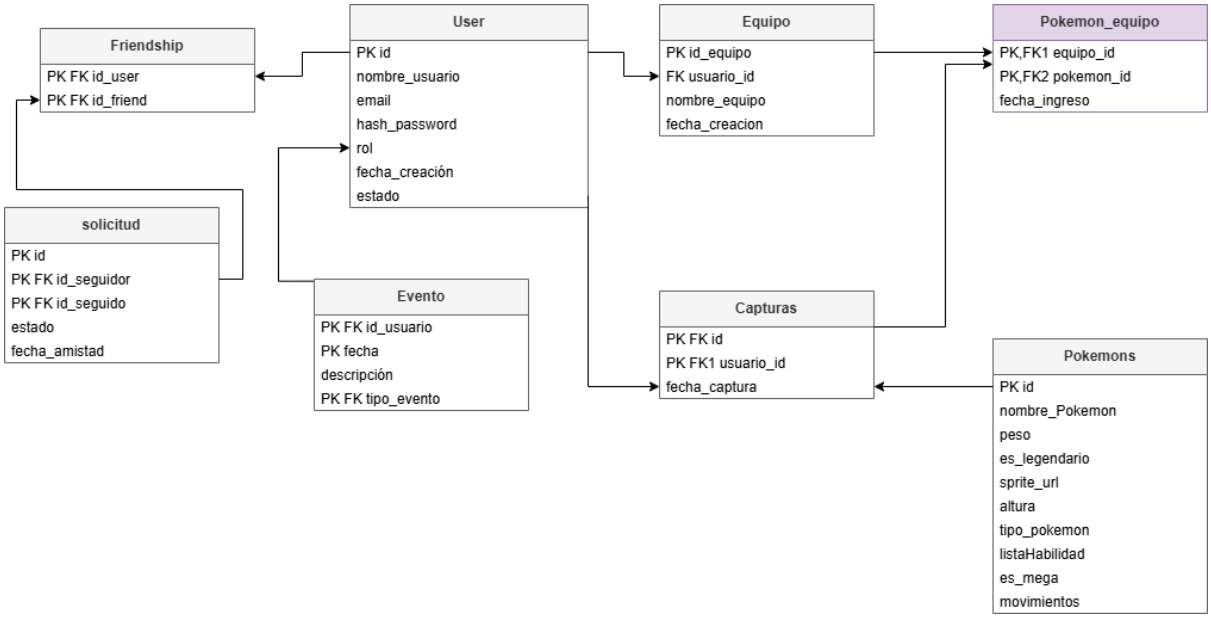

Figura 5.1: Esquema relacional BBDD

Capítulo 6

Diagramas de Comunicación

6.1. Introducción

En este capítulo se detallan los diagramas de comunicación que modelan la interacción entre los objetos del sistema.

6.2. Gestión de Usuarios (Punto 1)

6.2.1. Inicio Sesión

Figura 6.1: Diagrama de Comunicación: Inicio Sesión

Flujo de Mensajes:

1: El usuario introduce sus credenciales (usuario y contraseña) en la vista de Login y pulsa Ïniciar Sesión"

2: post_login(request)

2.1: identificarse (user, pass): Usuario

2.1.1: select()

2.1.2: new(): Usuario

6.2.2. Registro

Figura 6.2: Diagrama de Comunicación: Registro

Flujo de Mensajes:

1: El usuario introduce sus datos en la vista de registro y pulsa Registrarse"

2: post_registro(request)

2.1: registrarse(datos)

2.1.1: insert(datos)

2.1.2: new(): Usuario

6.2.3. Aprobar Cuenta

Figura 6.3: Diagrama de Comunicación: Aprobar Cuenta

Flujo de Mensajes:

1: El Administrador selecciona pendiente y click ."Aprobar"

2: post_aprobar_usuario(id)

2.1: aprobarCuenta(id)

2.1.1: select(id)

2.1.2: set_estado("Activo")

2.1.3: update()

6.2.4. Eliminar Cuenta

Figura 6.4: Diagrama de Comunicación: Eliminar Cuenta

Flujo de Mensajes:

1: El Administrador selecciona cuenta y click ."Eliminar"

2: post_eliminar_usuario(id)

2.1: eliminarCuenta(id)

2.1.1: select(id)

2.1.2: delete()

6.2.5. Enviar Solicitud

Figura 6.5: Diagrama de Comunicación: Enviar Solicitud

Flujo de Mensajes:

1: El usuario introduce id y click .enviar"

2: post_enviar_solicitud()

2.1: enviarSolicitud(idDestino)

2.2.1: select(idDestino)

2.2.2: new(estado = "pendiente")

2.2.3: insert()

6.2.6. Aceptar Solicitud

Figura 6.6: Diagrama de Comunicación: Aceptar Solicitud

Flujo de Mensajes:

1: El usuario pulsa aceptar en pendientes

2: post_aceptar(id)

2.1: aceptarSolicitud(id)

2.2.1: select(IdDestino)

2.2.2: set_estado(estado = .aceptada")

2.2.3: update()

6.2.7. Actualizar Datos Perfil

Figura 6.7: Diagrama de Comunicación: Actualizar Datos Perfil

Flujo de Mensajes:

1: El usuario modifica datos y pulsa "Guardar"

2: post_actualizar_perfil

2.1: actualizarDatos(datos)

2.1.1: select(id)

2.1.2: set_datos(nuevos_Datos)

2.1.3: update()

6.3. Gestión de Equipos Pokémon (Punto 2)

6.3.1. Consultar Equipo

Figura 6.8: Diagrama de Comunicación: Consultar Equipo

Flujo de Mensajes:

1: pulsar "Ver Equipo"

2: consultarEquipo(idEquipo)

2.1: obtener Datos Equipo (idEquipo)

2.2: buscarEquipo PorID (idEquipo)

2.2.1: execSQL("SELECT FROM Equipo WHERE id = %idEquipo %"): ResultadoSQL

2.2.1.1: new ResultadoSQL()

2.2.2: next(): boolean

2.3: buscar Pokemons DeEquipo(idEquipo)

2.3.1: execSQL("SELECT P.* FROM Pokemon P JOIN Pokemon_Equipo PE ON P.id=PE.pokemon_id WHERE PE.equipo_id= %id %"): ResultadoSQL

2.3.1.1: new ResultadoSQL()

6.3.2. Crear Equipo

Figura 6.9: Diagrama de Comunicación: Crear Equipo

Flujo de Mensajes:

1: confirmar(nombre, listaPokemons)

2: crearEquipo(idUser, nombre, listaPokemons)

2.1: validarTamaño(listaPokemons) : Boolean

2.2: execSQL(ÏNSERT INTO Equipo (user_id, nombre, fecha) VALUES (%idUser %, %nombre %, NOW())")

2.3: execSQL("SELECT MAX(id) as nuevo_id FROM Equipo WHERE user_id = %idUser %") : ResultadoSQL

2.3.1: new ResultadoSQL()

2.3.2: next() : boolean

2.3.3: getInt("nuevo_id") : int ->(Obtenemos idNuevoEquipo)

2.4: [Bucle] iterar listaPokemons

2.5: *[para cada idPk] execSQL(ÏNSERT INTO Pokemon_Equipo (equipo_id, pokemon_id) VALUES (%idNuevoEquipo %, %idPk %)")

2.6: registrarEvento(idUser, Creacion Equipo")

6.4. ChangeLog (Punto 3)

6.4.1. Registrar Evento (Guardar Equipo)

Figura 6.10: Diagrama de Comunicación: Registrar Evento

Flujo de Mensajes:

1: pulsar "Guardar Equipo"

2: crearEquipo(datosEquipo)

2.1: guardarNuevoEquipo(datosEquipo)

2.2: registrarEvento(idUsuario, tipoEvento, descripcion)

2.2.1: execSQL("SELECT 1 FROM TiposEvento WHERE tipo = %tipoEvento %"): ResultadoSQL

2.2.1.1: new ResultadoSQL()

2.2.2: next(): boolean

2.2.3: [existe] execSQL(ÏNSERT INTO Changelog_Event (idUsuario, tipo, descripcion, fecha) VALUES (%idUsuario %, %tipoEvento %, %descripcion %, NOW())")

6.4.2. Filtrar Notificaciones y Cargar Amigos

Figura 6.11: Diagrama de Comunicación: Filtrar Notificaciones

Flujo de Mensajes:

1: solicitarFiltroAmigos()

2: cargarListaAmigos(idUsuarioActual)

2.1: obtenerListaSeguidos(idUsuarioActual): List

2.1.1: execSQL("SELECT idSeguido FROM Seguidores WHERE idSeguidor = %idUsuarioActual %"): ResultadoSQL

2.1.1.1: new ResultadoSQL()

2.1.2: next(): boolean

2.1.3: getString(idSeguido): String

3: seleccionarUsuarioYBuscar(idAmigoSeleccionado)

4: cargarNotificaciones(listaIDs)

4.1: obtenerNotificaciones(listaIDs): List

4.1.1: execSQL("SELECT * FROM Changelog_Event WHERE idUsuario IN (%listaIDs %) ORDER BY fechaHora DESC"): ResultadoSQL

4.1.1.1: new ResultadoSQL()

4.1.2: next(): boolean

4.1.3: getString("descripcion"): String

6.5. Lista Completa de Pokémon y Búsquedas (Punto 4)

6.5.1. Filtros de Búsqueda

Figura 6.12: Diagrama de Comunicación: Filtros de Búsqueda

Flujo de Mensajes:

1: El usuario aplica los filtros que quiera para buscar uno o varios pokemons

2: getPokemons(filtros): List<Pokemon>

2.1: procesar_peticion(filtros): List<Pokemon>

2.1.1: getPokemons_Filtrados(filtros)

2.1.1.1: select()

2.1.1.2: new(): Pokemon

Capítulo 7

Changelog de la Documentación

Hemos añadido el modelo de dominio a los casos de uso y revisado la precondición de ver datos del administrador del sistema, para realizar el plan de pruebas hemos utilizado Gemini.

30/11 Hemos corregido el Modelo de Dominio y hemos reescrito los flujos de evento para realizar los diagrama de comunicación.

También hemos añadido el esquema relacional de la BBDD y el diagrama de clases.

La repartición del trabajo de esta entrega ha sido la siguiente: Cada integrante ha hecho los diagramas de comunicación correspondientes a su funcionalidad asignada:

Urko: funcionalidad 4, relacionada con la API de los pokemon.
Miguel: funcionalidad 1, relacionada con la gestión de usuarios.
Adrián: funcionalidad 3, relacionada con la ChangeLog
Maira: funcionalidad 2, relacionada con la gestión de equipos

La repartición respecto a la anterior entrega se ha enviado por correo.

Capítulo 8

Problemas de Implementación

8.1. Gestión de Usuarios

Uno de los mayores desafíos que encontré fue configurar la navegación inicial. Me costó bastante lograr que la pantalla de Login fuera obligatoriamente la primera ventana visible y que el sistema no intentase cargar el panel principal hasta que la sesión estuviera validada. Por otro lado, la implementación del buscador de usuarios fue lo más complejo a nivel de lógica. Tuve que desarrollar un sistema de filtrado que sirviera para dos propósitos muy distintos: permitir a los usuarios buscar nuevos amigos (filtrando los que ya lo eran) y permitir al Administrador localizar usuarios rápidamente para gestionar sus roles (ascenderlos o degradarlos) y permisos.

8.2. Crear equipos Pokémon

Mi mayor problema ha sido a la hora de juntar mi parte logica con la visual conjunta, ya que no me detectaba el botón, .Agregar Equipo ", pero una vez resuelto no he tenido mayor dificultad. Otro problema detectado, fue que al pulsar el botón .Editar Equipo ", no me abría correctamente la interfaz de Editar Equipo, sino que abria la de Consultar, un problema bastante sencillo de solucionar ya que era cambiar un bloque de código de la funcion definida para abrir dicha interfaz.

8.3. ChangeLog

Lo que mas me costó fue plantear la logica para que cada usuario viera solo lo que le corresponde, tuve bastantes problemas al principio con la consulta a la base de datos para filtrar por IDs. Ademas, se tardo bastante en implementar el filtro por fecha en la vista, ya que no conseguia que la lista se refrescara bien al seleccionar los dias y tuve que darle muchas vueltas al codigo hasta que logré que funcionara fluido junto con el resto de filtros

8.4. Lista Completa de Pokémon y Búsquedas con Filtros

Mi mayor problema a la hora de implementarlo ha sido la forma de tratar los datos al usar diferentes estructuras para la información, ademas de ello la parte de mostrar por la vista los datos obtenidos de los diferentes controladores también me ha costado pillarlo ya que el conocimiento inicial que tenia de Front era poca.

Capítulo 9

Conclusiones

Para ser sinceros, el proyecto nos ha dado más guerra de la que esperábamos. Al principio lo ves sobre el papel y dices 'vale, esto sale', pero cuando te pones a picar código y a intentar que todo funcione a la vez, la cosa cambia bastante.

Lo que más nos ha curtido ha sido:

Salir del entorno controlado: No es lo mismo hacer un ejercicio de clase que tener que pelearte con una API externa como la PokeAPI. Traerse miles de datos reales, filtrarlos y que no te explote la aplicación ha sido un reto, pero ahí es donde hemos espabilado de verdad.

Que todo cuadre: Más que la base de datos (que también tuvo lo suyo), lo difícil fue hacer que las piezas encajaran. Arreglabas una cosa en el código y de repente dejaba de ir otra que ni sabías que estaba relacionada. Ahí nos dimos cuenta de que programar es ser ordenado o morir en el intento.

Al final, creemos que ha sido un proyecto muy completo y el complemento perfecto para la asignatura. Una cosa es ver la teoría en las diapositivas y otra muy distinta es tener el error delante y tener que buscarte la vida para solucionarlo. Ver que ahora la aplicación tira de verdad y que puedes interactuar con todo, motiva bastante.

Capítulo 10

Repositorio

https://github.com/MairaHerbas/ADSIProyecto.git