Diseño arquitectónico de una Aplicación

Sirius

Protectora animal

Selene González Curbelo
Mariana Surós Álvarez

08/10/2023

ULPGC
Escuela de Ingeniería Informática

Índice

1. Definición del dominio

1.1 Elección del dominio

La aplicación móvil propuesta tiene como objetivo principal facilitar la gestión y promoción de la adopción y apadrinamiento de animales de una protectora. Está diseñada para ser utilizada tanto por los trabajadores de la protectora como por usuarios normales interesados en apoyar esta causa.

Objetivos de la aplicación:

- Facilitar la adopción y apadrinamiento de animales.
- Fomentar la participación y colaboración de voluntarios.
- Proporcionar información actualizada sobre noticias y eventos de la protectora.
- Optimizar el proceso de registro de los animales y sus datos clínicos.
- Agilizar el papeleo administrativo de la protectora.
- Establecer un canal de comunicación efectivo entre la protectora y los usuarios.
- Mejorar la visibilidad de la protectora y sus actividades.

1.2 Definición de principales funcionalidades

- Gestión de usuarios:
    - Registro de cuentas para diferentes roles: trabajador, administrador, dueño/a de la protectora, voluntario, usuario estándar, sin registrar.
    - Perfiles personalizados con información relevante para cada rol.
- Explorar animales:
    - Galería de fotos y detalles de animales disponibles para apadrinamiento o adopción.
    - Filtros de búsqueda por especie, estado (apadrinamiento, en adopción), etc.
- Perfil individual animal:
    - Detalles completos de un animal específico, incluyendo, nombre, edad, estado y descripción.
    - Opciones para apadrinar o adoptar.
- Registro clínico (para trabajadores):
    - Acceso exclusivo para ingresar y actualizar información médica de los animales.
    - Historial de tratamientos, medicamentos y diagnósticos.
- Gestión de eventos y noticias:
    - Publicación y actualización de eventos y noticias relevantes para la protectora y sus actividades.
- Comunicación entre roles:
    - Mensajería interna para facilitar la coordinación entre los implicados en la protectora.
    - Notificaciones para peticiones y actualizaciones.
    - Notificaciones sobre cambios en el estado de los animales (apadrinado, en adopción).
- Administración de facturas (para dueño/a):
    - Visualización y descarga de facturas asociadas a apadrinamientos, adopciones, gastos en el albergue.
- Niveles de acceso para voluntarios:

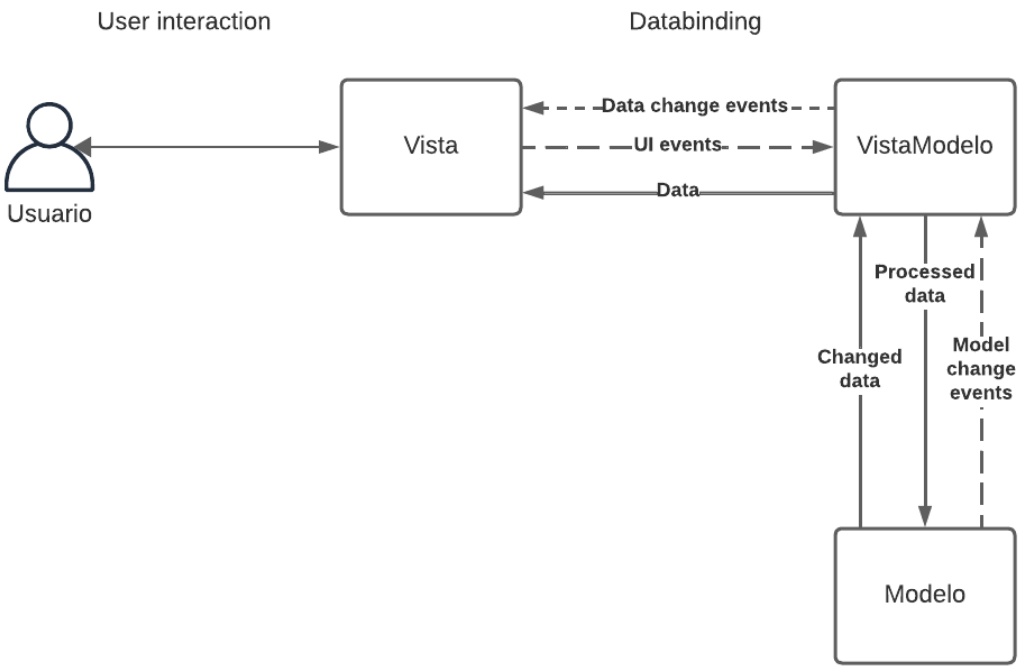
    - Distintos permisos basados en responsabilidades asignadas.
- Proceso de apadrinamiento y adopción:
    - Flujo guiado para completar solicitudes de apadrinamiento o adopción.
- Calendario de disponibilidad de voluntarios:
    - Para que los voluntarios puedan indicar su disponibilidad y ser asignados a tareas de acuerdo con sus horarios.
- Donaciones:
    - Sistema de donaciones y pagos en líneas para apoyar financieramente a la protectora.
- Idiomas adicionales:
    - Ofrecer la opción de cambiar el idioma de la aplicación.

2. Identificación de los principales elementos de la arquitectura elegida

La arquitectura seleccionada para esta aplicación es el Modelo-Vista-VistaModelo (MVVM). Esta elección se basa en la necesidad de separar de manera eficiente la lógica de presentación de la lógica de negocio, lo que permitirá una mayor modularidad y facilidad de mantenimiento. A continuación, desglosamos los principales componentes de la arquitectura MVVM:

- Modelo (*Backend*):
    - Representa la capa de datos y lógica de negocio de la aplicación.
    - Contiene clases y estructuras que definen cómo se almacenan, acceden y procesan los datos, incluyendo la gestión de usuarios, información de los animales, registros clínicos, eventos, noticias y funciones de comunicación entre roles.
- Vista (*Frontend*):
    - La Vista es la interfaz de usuario de la aplicación que los usuarios ven y con la que interactúan.
    - Contiene componentes visuales como botones, listas y formularios que permiten a los usuarios explorar galerías de animales, ver perfiles individuales de animales, completar formularios de registro de usuarios y animales, y acceder a páginas de eventos y noticias, entre otros.
- VistaModelo (Intermediario):
    - El *ViewModel* actúa como intermediario entre el Modelo y la Vista.
    - Gestiona las interacciones del usuario, procesa las solicitudes y controla la lógica de flujo de la aplicación.
    - En esta aplicación, los *ViewModels* se encargan de la lógica para la gestión de usuarios, búsqueda y visualización de animales, comunicación entre roles, proceso de apadrinamiento y adopción, y administración de eventos y noticias, entre otras funcionalidades.

3. Diseño de la arquitectura



4. Casos de uso

4.1. Caso de uso 1: Registro de un nuevo usuario

1. El usuario interactúa con la Vista (*Fronted*) para acceder al formulario de registro.
2. La Vista muestra un formulario de registro, que incluye campos como: nombre, dirección de correo electrónico y contraseña.
3. El usuario completa el formulario y presiona el botón de "Registrarse".
4. La Vista captura los datos proporcionados por el usuario y envía una solicitud al *ViewModel* correspondiente, indicando que se debe registrar un nuevo usuario.
5. El ViewModel procesa la solicitud, verifica los datos ingresados por el usuario y crea una nueva entrada de usuario en el Modelo, asegurando que cumpla con los requisitos de validación.
6. Una vez validados, el *ViewModel* para los datos al Modelo (*Backend*) para crear un nuevo usuario.
7. El Modelo almacena la información del nuevo usuario en la base de datos.
8. Una vez completado el registro, el *ViewModel* notifica a la Vista que la operación ha tenido éxito.
9. La Vista muestra un mensaje de confirmación al usuario y lo redirige a la página principal de la aplicación.

4.2. Caso de uso 2: Búsqueda y visualización de animales para adopción

1. El usuario interactúa con la Vista (*Fronted*) para acceder a la opción "Buscar animales en adoptación".
2. La Vista envía una solicitud de búsqueda al *ViewModel* correspondiente, con parámetros de filtrado (por ejemplo: tipo de animal, edad, tamaño, etc).
3. El *ViewModel* recibe la solicitud y procesa los criterios de búsqueda.

4. El *ViewModel* se comunica con el Modelo para obtener una lista de animales que coinciden con los criterios de búsqueda.
5. El Modelo consulta la base de datos y recopila la información de los animales disponibles, como fotos, nombres, edades y descripciones.
6. El *ViewModel* devuelve los datos recopilados a la Vista.
7. La Vista presenta la información de los animales en una galería o lista, que el usuario puede explorar.
8. Cuando el usuario selecciona un animal específico, la Vista muestra su perfil individual, incluyendo detalles adicionales.
9. Si el usuario decide apadrinar o adoptar a un animal, la Vista le permite iniciar el proceso correspondiente.

5. Conclusión

La arquitectura MVVM (Modelo-Vista-VistaModelo) es la base de nuestra aplicación de gran envergadura. Permite una separación eficiente entre la lógica de presentación y la lógica de negocio, esencial dada la magnitud y complejidad del proyecto. Proporciona modularidad y facilidad de mantenimiento, junto con un enlace de datos dinámico que mejora la experiencia del usuario móvil.

MVVM ofrece ventajas clave. Facilita la escalabilidad y la adición de nuevas funciones, gracias al mecanismo de Databinding que actualiza automáticamente la interfaz de usuario. Asimismo, promueve pruebas efectivas y un mantenimiento eficiente al asignar a cada componente una funcionalidad específica sin acoplamientos innecesarios.

Otras arquitecturas como MVC, MVP, MVI y Hexagonal fueron evaluadas. Aunque tienen méritos en diferentes contextos, no se alinean completamente con los objetivos y la complejidad de nuestra aplicación. Por ejemplo, MVC puede llevar a una mezcla de lógica en el Controlador, dificultando el mantenimiento y escalabilidad. MVP, aunque efectivo, puede generar complejidad adicional al requerir interfaces y presentadores para cada Vista, lo que sobrecargaría nuestra aplicación dada su envergadura.

MVI, aunque eficaz para aplicaciones altamente interactivas, puede resultar excesivamente complejo para nuestra aplicación. La arquitectura Hexagonal, valiosa para proyectos con fuerte desacoplamiento entre capas, puede ser innecesariamente compleja para nuestra aplicación.

En resumen, MVVM cumple con los objetivos de modularidad, mantenibilidad y eficiencia en la interfaz de usuario. Confiamos en que esta arquitectura nos permitirá alcanzar con éxito nuestros objetivos y brindar una experiencia excepcional a nuestros usuarios mientras desarrollamos la aplicación.