TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE CD. VALLES

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Desarrollo de aplicaciones móviles multiplataforma

Proyecto Wellness

NOMBRE DE ALUMNOS:

Castillo Gutiérrez Dara Madai - 21690340

Hernández Acuña Luis Eduardo – 21690128

Flores Pantoja Rubén Alberto – 20690544

Martínez Azuara José Luis - 20690372

DOCENTE:

Barrón Rodríguez Alfredo

Índice

Introducción

El presente documento describe el desarrollo de una aplicación móvil creada con Flutter, orientada al bienestar personal. La app, llamada “Wellness App”, busca ofrecer al usuario una experiencia integral para el seguimiento y mejora de su salud física y hábitos diarios. A través de una interfaz amigable e intuitiva, el usuario puede acceder a secciones como ejercicios, alimentación, progreso, suplementos, sueño, entre otras funcionalidades relevantes.

El desarrollo de esta aplicación se realizó con el objetivo de aplicar conocimientos adquiridos en el área de programación móvil, utilizando herramientas modernas como Flutter y SQLite para la gestión local de datos. La organización modular del proyecto facilita su mantenimiento y escalabilidad, permitiendo una navegación fluida y una experiencia de usuario coherente.

Objetivos

- Facilitar el acceso a rutinas de ejercicio con imágenes animadas (GIFs).

- Permitir el seguimiento del progreso del usuario.

- Ofrecer funcionalidades para registrar hábitos de alimentación, sueño y suplementos.

- Fomentar el bienestar integral de los usuarios.

Desarrollo

La app está estructurada en pantallas que permiten al usuario interactuar con diferentes secciones: ejercicios, alimentación, progreso, perfil, configuración, entre otras. Los datos son almacenados localmente utilizando una base de datos integrada. La carpeta 'assets' contiene imágenes animadas de los ejercicios, y el código Dart está organizado en módulos como 'screens', 'utils', y 'db'.

El archivo main.dart actúa como punto de entrada de la aplicación y se encarga de la navegación principal.

1. Pantalla de Inicio de Sesión (LoginScreen)

Funcionalidad: Esta pantalla sirve como punto de entrada para los usuarios existentes, permitiéndoles iniciar sesión con su nombre de usuario y contraseña. También proporciona navegación a la pantalla de registro para nuevos usuarios.

Explicación del Código:

El método _login maneja la autenticación del usuario. Recupera el usuario y la password introducidos de los TextEditingController.

Luego consulta la base de datos local SQLite utilizando _databaseHelper.getUser(usuario, password) para verificar las credenciales.

Al iniciar sesión con éxito, el usuario es navegado a la MainNavigationScreen, pasando los datos del usuario autenticado.

Los mensajes de error (por ejemplo, campos vacíos, credenciales incorrectas) se muestran utilizando _showSnackBar.

El booleano _obscurePassword y su IconButton asociado permiten alternar la visibilidad del campo de entrada de la contraseña.

2. Pantalla de Registro (SignUpScreen)

Funcionalidad: Esta pantalla permite a los nuevos usuarios crear una cuenta proporcionando datos personales como nombre de usuario, correo electrónico, contraseña, nombre, apellido, teléfono y fecha de nacimiento.

Explicación del Código:

El método _registerUser recoge datos de múltiples TextEditingController.

Realiza comprobaciones de validación para campos vacíos y coincidencia de contraseñas.

Antes de la inserción, comprueba la existencia del usuario y email en la base de datos para evitar entradas duplicadas utilizando _databaseHelper.usuarioExists y _databaseHelper.emailExists.

Si las validaciones pasan y no se encuentran duplicados, se llama a _databaseHelper.insertUser para añadir el nuevo usuario a la tabla users.

Se implementa un método _selectDate utilizando showDatePicker para permitir a los usuarios seleccionar su fecha de nacimiento, formateándola como 'dd/MM/yyyy'.

3. Pantalla de Navegación Principal (MainNavigationScreen)

Funcionalidad: Esta pantalla funciona como el centro principal de la aplicación después de un inicio de sesión exitoso. Utiliza un BottomNavigationBar para navegar entre las secciones principales: Búsqueda (Inicio), Favoritos y Configuración.

Explicación del Código:

La variable _currentIndex gestiona la pestaña activa en el BottomNavigationBar.

La lista _screens contiene instancias de las diversas pantallas (SearchScreen, FavoritesScreen, SettingsScreen) que se muestran según el _currentIndex.

El objeto user se pasa a cada pantalla secundaria para garantizar que los datos específicos del usuario sean accesibles en toda la aplicación.

4. Pantalla de Búsqueda (SearchScreen)

Funcionalidad: Esta pantalla presenta una interfaz similar a un panel de control donde los usuarios pueden acceder a diferentes módulos de la aplicación, incluyendo Ejercicios, Comidas, Progreso, Sueño y Suplementos.

Explicación del Código:

Muestra un saludo personalizado al usuario que ha iniciado sesión.

_buildIconButton es un método auxiliar para crear botones de icono visualmente atractivos que navegan a sus respectivas pantallas (EjerciciosScreen, AlimentacionScreen, ProgresoScreen, SuenoScreen, SuplementosScreen) al ser pulsados.

5. Pantalla de Ejercicios (EjerciciosScreen)

Funcionalidad: Esta pantalla clasifica los ejercicios por grupo muscular (por ejemplo, Pecho, Espalda, Brazo, Piernas y Glúteos). Los usuarios pueden tocar un grupo muscular para ver una lista de ejercicios asociados.

Explicación del Código:

El mapa estático ejerciciosPorGrupo define los datos de los ejercicios, categorizándolos por grupo muscular e incluyendo su nombre, imagen (nombre del archivo GIF) y nota (descripción).

Los widgets ListTile se utilizan para mostrar cada grupo muscular, y al tocarlos se navega a EjerciciosGrupoScreen, pasando la lista de ejercicios relevante y el usuario actual.

6. Pantalla de Grupo de Ejercicios (EjerciciosGrupoScreen)

Funcionalidad: Esta pantalla muestra una lista de ejercicios pertenecientes a un grupo muscular específico. Los usuarios pueden ver los detalles de cada ejercicio (imagen, nota) y marcarlos/desmarcarlos como favoritos.

Explicación del Código:

El Set _favoritos almacena los nombres de los ejercicios marcados como favoritos para el usuario actual.

_cargarFavoritos recupera los ejercicios favoritos del usuario de la columna ejercicios_favoritos en la tabla users de la base de datos. También actualiza SharedPreferences para mantener la coherencia.

_guardarFavoritos persiste la lista actualizada de ejercicios favoritos tanto en SharedPreferences como en la base de datos.

_toggleFavorito añade o elimina un ejercicio del conjunto _favoritos y activa _guardarFavoritos.

Cuando se toca un ejercicio, showDialog muestra un AlertDialog con la imagen y la nota del ejercicio.

7. Pantalla de Alimentación (AlimentacionScreen)

Funcionalidad: Esta pantalla muestra una lista completa de alimentos, cada uno con información nutricional (objetivo, calorías, proteínas, grasas, carbohidratos, ingredientes). Los usuarios pueden marcar/desmarcar alimentos como favoritos.

Explicación del Código:

_favoritos es una lista que almacena los nombres de los alimentos favoritos para el usuario actual.

cargarFavoritos carga los alimentos favoritos del usuario desde SharedPreferences utilizando una clave específica del usuario (favoritos_comidas$usuario). Decodifica la

cadena JSON en una lista de índices y luego recupera los nombres de los alimentos correspondientes de comidasGlobal.

_toggleFavorito maneja la adición o eliminación de alimentos de _favoritos. Luego actualiza la columna comidas_favoritas en la base de datos utilizando dbHelper.updateComidasFavoritas y también actualiza SharedPreferences.

Los widgets ExpansionTile se utilizan para mostrar los detalles de los alimentos, permitiendo a los usuarios expandirlos para ver los ingredientes.

8. Pantalla de Progreso (ProgresoScreen)

Funcionalidad: Esta pantalla permite a los usuarios registrar su peso y altura a lo largo del tiempo. Muestra el progreso actual y un historial de mediciones anteriores.

Explicación del Código:

historialProgreso almacena una lista de mapas, donde cada mapa representa una entrada de progreso con peso, altura y fecha.

_loadProgreso recupera el historial de progreso del usuario de la columna progreso en la base de datos, manejando tanto el formato de lista como el de mapa único para compatibilidad con versiones anteriores.

_guardarProgreso añade una nueva entrada de progreso a historialProgreso (insertándola al principio para un orden cronológico inverso) y guarda el historial actualizado como una cadena JSON en la base de datos utilizando DatabaseHelper().updateProgreso.

Los widgets TextField se utilizan para introducir el peso y la altura actuales.

9. Pantalla de Sueño (SuenoScreen)

Funcionalidad: Esta pantalla permite a los usuarios registrar sus horas de sueño diarias y establecer un objetivo de sueño. Realiza un seguimiento del historial de sueño y muestra cómo el sueño del día actual se compara con el objetivo establecido.

Explicación del Código:

historialSueno almacena una lista de entradas de sueño, cada una con una fecha y las horas dormidas.

objetivoSueno almacena las horas de sueño objetivo del usuario.

_cargarDatos carga tanto el historial de sueño como el objetivo desde SharedPreferences, utilizando claves específicas del usuario (_claveHistorial y _claveObjetivo).

_guardarDatos persiste el historial de sueño y el objetivo actualizados en SharedPreferences.

_agregarRegistro añade o actualiza una entrada de sueño para el día actual.

_guardarObjetivo actualiza el objetivo de sueño del usuario.

La interfaz de usuario muestra el último sueño registrado, el objetivo de sueño, y una lista de entradas de sueño anteriores. También indica visualmente si se alcanzó el objetivo de sueño para la última entrada.

10. Pantalla de Suplementos (SuplementosScreen)

Funcionalidad: Esta pantalla proporciona información sobre varios suplementos. Cada suplemento tiene una descripción, una dosis recomendada y una lista de beneficios.

Explicación del Código:

La lista suplementos contiene datos codificados para cada suplemento, incluyendo nombre, descripcion, dosis y beneficios.

Los widgets ExpansionTile se utilizan para mostrar los detalles de cada suplemento, permitiendo a los usuarios expandirlos para ver la descripción, la dosis y los beneficios.

11. Pantalla de Favoritos (FavoritesScreen)

Funcionalidad: Esta pantalla actúa como una ubicación centralizada para que los usuarios vean sus ejercicios y alimentos favoritos.

Explicación del Código:

Proporciona dos widgets _favoriteCard, uno para "Ejercicios" y uno para "Comidas".

Al tocar en "Ejercicios" se navega a EjerciciosFavoritosScreen, y al tocar en "Comidas" se navega a ComidasFavoritasScreen.

12. Pantalla de Comidas Favoritas (ComidasFavoritasScreen)

Funcionalidad: Esta pantalla muestra una lista de los alimentos que el usuario ha marcado como favoritos. Los usuarios también pueden eliminar elementos de esta lista.

Explicación del Código:

loadComidasFavoritas recupera la lista de alimentos favoritos de SharedPreferences utilizando la clave específica del usuario (favoritos_comidas$usuario). Luego mapea los índices almacenados de nuevo a los datos de alimentos reales de comidasGlobal.

_eliminarDeFavoritos elimina un alimento específico de la lista de favoritos, actualiza SharedPreferences, y luego recarga la lista.

13. Pantalla de Ejercicios Favoritos (EjerciciosFavoritosScreen)

Funcionalidad: Esta pantalla muestra una lista de ejercicios que el usuario ha marcado como favoritos. Los usuarios pueden eliminar elementos de esta lista.

Explicación del Código:

_cargarFavoritos recupera los ejercicios favoritos del usuario de la columna ejercicios_favoritos en la base de datos.

_eliminarFavorito elimina un ejercicio de la lista favoritos y luego actualiza la columna ejercicios_favoritos en la base de datos utilizando DatabaseHelper().updateEjerciciosFavoritos.

14. Pantalla de Perfil (PerfilScreen)

Funcionalidad: Esta pantalla muestra la información personal del usuario (nombre de usuario, correo electrónico, nombre, teléfono, fecha de nacimiento) y le permite actualizar su foto de perfil y otros detalles editables.

Explicación del Código:

_profileImage almacena el archivo de imagen de perfil seleccionado. _pickImage utiliza image_picker para permitir a los usuarios seleccionar una imagen de su galería. La ruta de la imagen seleccionada se guarda luego en la base de datos utilizando DatabaseHelper().updateProfileImage.

_editarPerfil presenta un AlertDialog con campos de texto para editar el nombre, el correo electrónico y el teléfono.

Si el usuario confirma las ediciones, se llama a DatabaseHelper().updateUserData para persistir los cambios en la base de datos.

15. Pantalla de Cambio de Contraseña (ChangePasswordScreen)

Funcionalidad: Esta pantalla permite a los usuarios cambiar su contraseña proporcionando su correo electrónico actual, la contraseña antigua y una nueva contraseña (con confirmación).

Explicación del Código:

El método _changePassword recupera la entrada de los TextEditingController.

Valida que todos los campos estén rellenos y que las nuevas contraseñas coincidan.

Luego intenta recuperar al usuario con el correo electrónico y la contraseña antigua proporcionados utilizando _databaseHelper.getUser.

Si el usuario se encuentra y las credenciales son correctas, se llama a _databaseHelper.updatePassword para actualizar la contraseña en la base de datos.

_showSnackBar se utiliza para proporcionar retroalimentación al usuario sobre el éxito o el fracaso.

16. Pantalla de Configuración (SettingsScreen)

Funcionalidad: Esta pantalla ofrece opciones para gestionar la cuenta del usuario, incluyendo la navegación a la pantalla de edición de perfil, el cambio de contraseña y el cierre de sesión de la aplicación.

Explicación del Código:

Utiliza widgets ListTile para representar cada opción de configuración.

Al tocar en 'Perfil' se navega a PerfilScreen, y al tocar en 'Cambiar contraseña' se navega a ChangePasswordScreen.

La opción 'Cerrar sesión' utiliza Navigator.pushReplacementNamed(context, '/login') para cerrar la sesión del usuario y volver a la pantalla de inicio de sesión.

Ayudante de Base de Datos (DatabaseHelper)

Funcionalidad: Esta clase singleton proporciona una forma centralizada de interactuar con la base de datos SQLite, manejando la inicialización de la base de datos, la inserción de usuarios, la recuperación y varias operaciones de actualización de datos.

Explicación del Código:

Patrón Singleton: DatabaseHelper._instance y factory DatabaseHelper() aseguran que solo exista una instancia de DatabaseHelper en toda la aplicación.

_initDb(): Este método inicializa la base de datos SQLite. Determina la ruta de la base de datos utilizando getApplicationDocumentsDirectory() y abre la base de datos. Define el esquema de la tabla users, que incluye columnas para los detalles del usuario, la imagen de perfil, las comidas favoritas, los ejercicios favoritos y los datos de progreso.

insertUser(Map\<String, dynamic\> user): Inserta un nuevo registro de usuario en la tabla users.

getUser(String usuario, String password): Recupera un usuario de la base de datos basándose en su nombre de usuario y contraseña, utilizado para la autenticación de inicio de sesión.

emailExists(String email) y usuarioExists(String usuario): Estos métodos comprueban la existencia de un correo electrónico o nombre de usuario en la base de datos, lo que es crucial para evitar registros duplicados.

updatePassword(String email, String newPassword): Actualiza la contraseña de un usuario basándose en su correo electrónico.

updateUserData(...): Actualiza el nombre, el correo electrónico y el número de teléfono de un usuario.

updateProfileImage(String email, String imagePath): Actualiza la ruta de la imagen de perfil del usuario.

updateComidasFavoritas(dynamic usuario, List\<String\> favoritos): Actualiza la columna comidas_favoritas, almacenando la lista de alimentos favoritos como una cadena JSON.

updateEjerciciosFavoritos(String usuario, List\<String\> ejercicios): Actualiza la columna ejercicios_favoritos, almacenando la lista de ejercicios favoritos como una cadena JSON.

updateProgreso(String usuario, String progresoJson): Actualiza la columna progreso, almacenando el historial de progreso del usuario como una cadena JSON.

Datos Globales y Utilidades

comidas_global.dart (Implícito en AlimentacionScreen y ComidasFavoritasScreen):

Aunque no se proporciona como un archivo separado en el código dado, la presencia y el uso de comidasGlobal en AlimentacionScreen y ComidasFavoritasScreen implican un archivo llamado comidas_global.dart que contiene una lista global de alimentos con su información nutricional. Es probable que estos datos estén estructurados como una List\<Map\<String, dynamic\>\>.

shared_preferences (para AlimentacionScreen, EjerciciosGrupoScreen, SuenoScreen):

Se utiliza para el almacenamiento ligero de pares clave-valor, principalmente para almacenar en caché datos específicos del usuario, como los alimentos favoritos, los ejercicios favoritos y el historial de sueño. Esto ayuda a cargar rápidamente las preferencias del usuario sin tener que consultar siempre la base de datos para los datos a los que se accede con frecuencia.

path_provider (para DatabaseHelper):

Proporciona acceso a ubicaciones de uso común en el sistema de archivos, esencial para determinar dónde almacenar el archivo de la base de datos SQLite.

sqflite (para DatabaseHelper):

El plugin oficial de SQLite para Flutter, utilizado para operaciones de base de datos como la creación de tablas, la inserción, la consulta y la actualización de datos.

path (para DatabaseHelper):

Un paquete de utilidad para manipular rutas, utilizado en DatabaseHelper para unir rutas de directorio y construir la ruta del archivo de la base de datos.

dart:convert (para DatabaseHelper, AlimentacionScreen, ComidasFavoritasScreen, EjerciciosGrupoScreen, EjerciciosFavoritosScreen, ProgresoScreen, SuenoScreen):

Se utiliza para codificar y decodificar datos a y desde el formato JSON, particularmente para almacenar listas o mapas en una sola columna TEXT dentro de la base de datos SQLite (por ejemplo, comidas_favoritas, ejercicios_favoritos, progreso) y para SharedPreferences.

image_picker (para PerfilScreen):

Un plugin de Flutter para seleccionar imágenes de la galería de imágenes o la cámara, utilizado para permitir a los usuarios seleccionar y actualizar su foto de perfil.

intl (para SignUpScreen, SuenoScreen):

Se utiliza para la internacionalización, específicamente para formatear fechas en SignUpScreen (DateFormat('dd/MM/yyyy').format(picked)) y en SuenoScreen para mostrar la fecha actual (DateFormat('dd/MM/yyyy').format(DateTime.now())).

Conclusión

Proyecto Wellness se consolida como una herramienta integral y accesible, diseñada para potenciar el bienestar general de sus usuarios. Gracias a su interfaz intuitiva y a un conjunto de funcionalidades específicas, la aplicación permite un control detallado y efectivo de rutinas y hábitos saludables.

La integración de módulos clave como el seguimiento de ejercicios, la gestión de la alimentación, el registro del progreso físico, el monitoreo del sueño y la información sobre suplementos, facilita una aproximación holística al bienestar. Esto empodera al usuario para tomar decisiones informadas y personalizar su camino hacia una vida más sana.

En resumen, Proyecto Wellness no es solo una aplicación, sino un valioso compañero digital que fomenta la autodisciplina y apoya activamente la consecución de objetivos de salud de manera sostenible y significativa.

Bibliografías

- Documentación oficial de Flutter: https://flutter.dev/docs

- Documentación de SQLite en Flutter

- Recursos de diseño y desarrollo móvil utilizados durante el proyecto