Lo que debará hacer la aplicación en una primera versión:

1. **Recoge datos manuales** del usuario (edad, peso, altura, BPM, tipo de entrenamiento…).
2. **Calcula nuevas variables** (como BMI y Log\_Age) con el mismo proceso de Feature Engineering del entrenamiento.
3. **Aplica el modelo de predicción** entrenado previamente.
4. **Guarda la predicción y los datos originales** en el Feature Store local, junto con el email del usuario, un user\_id correlativo y un timestamp.
5. **(Opcional)** Si el usuario introduce su grasa corporal real, también se guarda.
6. **(Opcional)** Puede mostrar al usuario su evolución si ya ha introducido datos antes.

**Estructura del Feature Store**

Cada fila guardada tendrá estas columnas:

| user\_id | email | timestamp | edad | peso | altura | ... | BMI | predicción | real (opcional) |

**Concepto General del Sistema**

1. **Entrada manual de datos por el usuario**
   * A través de una interfaz Streamlit.
   * Introduce los mismos campos que el dataset original.
2. **Preprocesamiento**
   * Calculamos campos como BMI y Log\_Age con el mismo feature\_engineering ya implementado.
3. **Predicción**
   * Usamos el modelo entrenado (Random Forest) para predecir la grasa corporal.
4. **Guardado en Feature Store**
   * Guardamos los datos introducidos, la predicción y el valor real (si se introduce).
   * En un único archivo feature\_store/user\_fat\_percentage/features.csv.
   * Si ya existe una entrada con el mismo correo y fecha, la sobrescribimos.
5. **(Opcional en el futuro)**: Retrain automático del modelo con los nuevos datos añadidos.

**1/06/2025**

**✅ 1. Estructura del Proyecto**

**Qué hicimos**:

* Organizamos los notebooks (01\_ a 08\_) y los scripts de código en src/ correctamente.
* Consolidamos los datos en carpetas estándar: data/raw, data/interim, data/processed.
* Centralizamos el feature\_store en la raíz, con rutas versionadas (v1\_YYYY-MM-DD).

**Ventajas**:

* Mejor mantenimiento del proyecto y escalabilidad.
* Evita duplicidades y errores por archivos fuera de sitio.
* Cumple con buenas prácticas de MLops y estructura reproducible.

**✅ 2. Preprocesamiento y Feature Engineering**

**Qué hicimos**:

* Refactorizamos los notebooks 02\_ y 03\_ para detectar automáticamente la raíz del proyecto.
* Separaste la lógica de feature engineering y preprocesamiento en scripts reutilizables:
  + build\_features.py
  + preprocessing.py
  + preprocess\_and\_save.py

**Ventajas**:

* Código más limpio, reutilizable y testable.
* Permite usar los mismos pasos tanto en notebooks como en app y scripts.
* Flexibilidad para reentrenar o predecir con nuevos datos fácilmente.

**✅ 3. Feature Store Local**

**Qué hicimos**:

* Implementamos un Feature Store local:
  + Centralizado: feature\_store/user\_fat\_percentage/v1/features.csv
  + Versión con fecha: feature\_store/user\_fat\_percentage/v1\_2024-06-01/features.csv
  + Script de utilidades: feature\_store\_utils.py con funciones:
    - save\_versioned\_dataset()
    - list\_versions()
    - track\_metadata()
    - validate\_schema()

**Ventajas**:

* Historial de versiones de datos disponible sin sobrescribir.
* Trazabilidad de cambios (fecha, columnas, tipos).
* Preparado para crecer a producción o Hopsworks/Feast en el futuro.

**✅ 4. Modelos y Pipelines**

**Qué hicimos**:

* Estandarizamos el guardado de modelos en models/.
* Creamos dos scripts separados:
  + train\_pipeline.py → Pipeline con hiperparámetros por defecto.
  + train\_pipeline\_optuna.py → Pipeline optimizado con Optuna.

**Ventajas**:

* Control total de modelos.
* Facilita comparar modelos (base vs optimizado).
* Mejora precisión del modelo gracias a la optimización.

**✅ 5. Preparado para la app Streamlit**

**Qué dejamos preparado**:

* Confirmamos que los datos introducidos por usuarios pasarán por el mismo proceso que los datos originales (feature engineering y preprocesamiento).
* Definimos el comportamiento deseado:
  + Guardar predicción (y grasa real si se proporciona).
  + Un único archivo CSV con email, fecha, predicción, etc.
  + Reemplazo si ya existe una fila con ese email + fecha.

**Ventajas**:

* Trazabilidad por usuario.
* El modelo podrá reentrenarse con datos reales introducidos por los propios usuarios.
* Fomentamos la mejora continua de la predicción de grasa corporal.

**✅ 6. Validación del entorno**

**Qué hicimos**:

* Solucionamos errores de entorno (conflictos con ipykernel, instalación fallida de hopsworks, errores con twofish, etc.).
* Confirmamos que todos los notebooks funcionan correctamente.

**Ventajas**:

* Entorno estable y reproducible con poetry.
* Posibilidad de ejecutar notebooks, scripts o apps sin errores inesperados.

**Archivos .py que tienes ahora y para qué sirve cada uno**

**🔷 src/data/make\_dataset.py**

**👉 Qué hace:** carga datos **raw** y los guarda en data/interim/ después de limpieza básica (si se necesita).  
**💡 Por qué es útil:** centraliza la carga de datos en un solo lugar. Así si cambia la fuente, lo actualizas aquí y listo.

**🔷 src/features/build\_features.py**

**👉 Qué hace:** crea nuevas columnas derivadas (como BMI y Log\_Age) a partir de los datos originales.  
**💡 Por qué es útil:** permite que el mismo cálculo se aplique siempre de forma consistente, también en la app.

**🔷 src/features/preprocessing.py**

**👉 Qué hace:** escala las columnas numéricas y codifica las categóricas (con StandardScaler y LabelEncoder).  
**💡 Por qué es útil:** transforma los datos antes de entrenar o predecir para que los modelos trabajen correctamente.

**🔷 src/features/feature\_store\_utils.py**

**👉 Qué hace:** guarda datasets en el feature\_store, con control de versiones, validación del schema y guardado de metadatos.  
**💡 Por qué es útil:** lleva el control de qué datos usaste en cada modelo y permite reproducir entrenamientos.

**🔷 src/feature\_store/utils.py**

**👉 Qué hace:** lo mismo que el anterior pero más genérico. Puedes usar uno u otro, pero deberías consolidarlo en uno solo.  
**💡 Recomendación:** usa **solo feature\_store\_utils.py** y elimina el duplicado si no lo necesitas.

**🔷 src/models/train\_pipeline.py**

**👉 Qué hace:** entrena un modelo con un pipeline completo (preprocesado + modelo) y lo guarda.  
**💡 Por qué es útil:** puedes lanzar un reentrenamiento sin abrir notebooks. Ideal para automatizar y reutilizar.

**🔷 src/models/train\_pipeline\_optuna.py**

**👉 Qué hace:** igual que el anterior pero usando Optuna para encontrar los mejores hiperparámetros.  
**💡 Por qué es útil:** obtienes un mejor modelo automáticamente. También es reproducible y guardado.

**🔷 src/models/predict\_model.py**

**👉 Qué hará (todavía vacío):** cargará el modelo y usará datos del usuario para hacer predicciones.  
**💡 Por qué es útil:** será usado por la app Streamlit para predecir grasa corporal.

**🔷 src/models/train\_model.py *(opcional)***

**👉 Qué haría:** si no usas train\_pipeline.py, aquí podrías tener entrenamiento separado (sin pipeline).  
**💡 Recomendación:** si ya tienes train\_pipeline.py, este podrías eliminarlo o dejarlo solo si usas otra técnica.

**🔷 src/visualization/visualize.py**

**👉 Qué hará:** código para mostrar gráficas (por ejemplo, evolución de grasa corporal por persona).  
**💡 Por qué es útil:** tener visualizaciones limpias que puedes reutilizar desde notebooks o app.

**✅ En resumen: estructura limpia y modular**

| **Categoría** | **Archivos clave** | **Para qué sirve** |
| --- | --- | --- |
| **Datos** | make\_dataset.py | Cargar y preparar datos crudos |
| **Features** | build\_features.py, preprocessing.py | Crear y transformar columnas |
| **Feature Store** | feature\_store\_utils.py | Guardar datasets con versión, validación y metadatos |
| **Modelos** | train\_pipeline.py, train\_pipeline\_optuna.py, predict\_model.py | Entrenar, optimizar y predecir modelos |
| **Visualización** | visualize.py | Gráficas reutilizables |

**Siguientes pasos a construir: 1/6/2025**

**PRÓXIMOS PASOS PROPUESTOS**

1. La app en Streamlit para introducir datos.
2. Guardar esos datos en el Feature Store.
3. Predecir grasa corporal y mostrar resultados.
4. Añadir entrada opcional de valor real para mejorar el modelo.
5. (Más adelante) permitir reentrenamiento automático con nuevos datos.
6. **(Opcional)** Añadir validación de esquema (para evitar errores).