**1. Definición Estratégica del Proyecto**

**Resumen del Proyecto**

* **Nombre de la Aplicación**: **PayRoll Pro**
* **Descripción**: PayRoll Pro es una aplicación web diseñada para automatizar y gestionar el cálculo de pagos adicionales y variables para un equipo de trabajo. El sistema permite registrar días trabajados, gestionar un calendario de días feriados anuales, y procesar la información de forma manual o mediante la importación masiva de archivos CSV para calcular automáticamente conceptos como dietas y taxis según reglas de negocio específicas.
* **Público Objetivo**: Líderes de equipo, supervisores o personal administrativo que actualmente gestionan estos cálculos de forma manual y necesitan una herramienta para centralizar la información, reducir errores y agilizar el proceso de reporte de nómina.
* **Objetivo Principal**: Simplificar y automatizar el cálculo de pagos variables (dietas, taxis, contingencias), eliminando la dependencia de hojas de cálculo y procesos manuales propensos a errores. La aplicación proporcionará un registro claro, auditable y flexible de las jornadas laborales especiales y los días feriados del año.

**Funcionalidades Clave**

* **MVP (Producto Mínimo Viable)**:
  + Módulo de autenticación de usuarios (login/logout).
  + Gestión de empleados (crear, editar), incluyendo su nombre, turno y día libre predeterminado.
  + **Módulo de Días Feriados**:
    - Formulario para registrar manualmente los días feriados del año (nombre y fecha).
    - Funcionalidad para importar un archivo CSV con la lista de feriados.
  + **Módulo de Registros Laborales**:
    - Formulario para registrar manualmente una jornada laboral especial (sexto día, contingencia, etc.) para un empleado.
    - Funcionalidad para importar un archivo CSV con múltiples registros de jornadas.
  + Cálculo automático de dietas y taxis basado en los registros existentes.
  + Una vista de resumen o reporte que muestre los totales calculados por empleado.
* **Mejoras Futuras**:
  + Generación de reportes históricos por mes o rango de fechas.
  + Exportación de reportes a formatos como PDF o Excel.
  + Roles de usuario (Administrador vs. Empleado que solo puede ver su información).
  + Un dashboard con estadísticas visuales.
  + Sincronización automática de días feriados desde una API externa.

Esta estrategia actualizada incluye la gestión de feriados como una pieza central del MVP, fortaleciendo la lógica de negocio desde el inicio.

**2. Arquitectura y Estructura de Apps Django**

La estructura de aplicaciones se mantiene robusta y adecuada para incorporar la nueva funcionalidad.

* **App core**: Contendrá la lógica y elementos transversales del proyecto, como la plantilla base.html y helpers globales.
* **App users**: Se encargará de la gestión de usuarios, extendiendo el modelo por defecto de Django para añadir campos como el turno y el día libre.
* **App payroll**: Será el corazón de la aplicación. Gestionará los modelos de Turnos, Tipos de Jornada, **Días Feriados**, y Registros Laborales. También contendrá toda la lógica de negocio para los cálculos y las importaciones de CSV.

La nueva funcionalidad encaja perfectamente dentro de las responsabilidades de la app payroll.

**3. Diseño Detallado de la Implementación**

**Modelo de Datos (models.py por App)**

**App users:**

* **Modelo User (extendiendo AbstractUser):**
  + **Campos**: employee\_id (CharField), shift (ForeignKey a payroll.Shift), default\_off\_day (IntegerField).

**App payroll:**

* **Modelo Shift:**
  + **Campos**: name (CharField), slug (SlugField).
* **Modelo WorkType:**
  + **Campos**: name (CharField), slug (SlugField).
* **Modelo Holiday (Nuevo):**
  + **Campos**:
    - name: CharField(max\_length=100) - Nombre de la festividad (ej: "Día de la Restauración").
    - date: DateField(unique=True) - La fecha específica del feriado.
  + **Método get\_absolute\_url**: Implementará la URL para ver o editar este registro.
* **Modelo WorkLog:**
  + **Campos**: employee (ForeignKey a users.User), work\_date (DateField), work\_type (ForeignKey a WorkType), alerts\_worked (PositiveIntegerField), apply\_taxi (BooleanField).
  + **Método get\_absolute\_url**: Implementará la URL para ver el detalle de este registro.

**Rutas y Lógica de Vistas (urls.py y views.py por App)**

* **App payroll**:
  + **urls.py**: Se definirá app\_name = 'payroll'.
  + **Vistas para WorkLog**: WorkLogListView, WorkLogDetailView, WorkLogCreateView, WorkLogUpdateView, WorkLogDeleteView, ImportCSVView.
  + **Vistas para Holiday (Nuevas)**:
    - HolidayListView (holiday\_index.html): Listará todos los feriados del año, con paginación.
    - HolidayCreateView (holiday\_create.html): Formulario para añadir un nuevo feriado.
    - HolidayUpdateView (holiday\_edit.html): Formulario para editar un feriado existente.
    - HolidayDeleteView (holiday\_confirm\_delete.html): Página de confirmación para eliminar.
    - ImportHolidaysCSVView: Vista con un formulario para la carga masiva de feriados desde un CSV.

**Formularios (forms.py por App)**

* **App payroll**:
  + WorkLogForm: ModelForm basado en el modelo WorkLog.
  + **HolidayForm (Nuevo)**: ModelForm basado en el modelo Holiday.

Python

# payroll/forms.py

from django import forms

from .models import WorkLog, Holiday

class HolidayForm(forms.ModelForm):

class Meta:

model = Holiday

fields = ['name', 'date']

widgets = {

'date': forms.DateInput(attrs={'type': 'date'})

}

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

common\_classes = 'mt-1 block w-full rounded-md border-slate-300 dark:border-slate-600 dark:bg-slate-800 shadow-sm focus:border-blue-500 focus:ring-blue-500 sm:text-sm'

for field\_name, field in self.fields.items():

field.widget.attrs.update({'class': common\_classes})

**Diseño de Plantillas y Frontend**

La estructura y diseño se mantienen, con la adición de las plantillas para el nuevo módulo.

* **Estructura de Templates**: Se crearán las plantillas para feriados: payroll/templates/payroll/holiday\_index.html, holiday\_create.html, etc.
* **Paleta de Clases (Tailwind CSS)**: Se mantiene la elección de **Blue** como color principal y **Slate** como neutro para consistencia visual.
* **Componentes Reutilizables (Partials)**:
  + \_navbar.html: Se añadirá un enlace en la navegación para acceder a la sección "Gestionar Feriados".
  + Se reutilizarán \_alerts.html, \_pagination.html y \_form.html para el nuevo módulo de feriados.

**4. Configuración, Documentación y Consideraciones Finales**

La configuración del proyecto no sufre cambios significativos.

* **Dependencias (requirements.txt)**: Se mantiene la lista base: django, python-decouple, psycopg2-binary, Pillow.
* **Configuraciones Clave (settings.py)**: Sin cambios. Se usará python-decouple y se registrarán las apps core, users y payroll.
* **Gestión de Secretos y Control de Versiones**: Se mantiene el uso de .env y un .gitignore completo.

**5. Datos de Prueba (SQL Seeder)**

El script seeds.sql se actualizará para incluir datos en la nueva tabla de feriados.

**Orden de Inserción (Respetando Foreign Keys)**:

1. payroll\_shift
2. payroll\_worktype
3. **payroll\_holiday (Nuevo)**
4. users\_user
5. payroll\_worklog

SQL

-- seeds.sql

-- Insertar Turnos (Shifts)

INSERT INTO "payroll\_shift" ("id", "name", "slug") VALUES (1, 'Turno Mañana', 'manana');

INSERT INTO "payroll\_shift" ("id", "name", "slug") VALUES (2, 'Turno Tarde', 'tarde');

INSERT INTO "payroll\_shift" ("id", "name", "slug") VALUES (3, 'Turno Amanecida', 'amanecida');

-- Insertar Tipos de Jornada (WorkTypes)

INSERT INTO "payroll\_worktype" ("id", "name", "slug") VALUES (1, 'Sexto Día', 'sexto-dia');

INSERT INTO "payroll\_worktype" ("id", "name", "slug") VALUES (2, 'Contingencia', 'contingencia');

INSERT INTO "payroll\_worktype" ("id", "name", "slug") VALUES (3, 'Contingencia Especial', 'contingencia-especial');

INSERT INTO "payroll\_worktype" ("id", "name", "slug") VALUES (4, 'Feriado', 'feriado');

INSERT INTO "payroll\_worktype" ("id", "name", "slug") VALUES (5, 'Día Regular', 'regular');

-- Insertar Días Feriados (Holidays)

INSERT INTO "payroll\_holiday" ("id", "name", "date") VALUES (1, 'Día del Trabajo', '2025-05-01');

INSERT INTO "payroll\_holiday" ("id", "name", "date") VALUES (2, 'Día de la Restauración', '2025-08-16');

INSERT INTO "payroll\_holiday" ("id", "name", "date") VALUES (3, 'Día de las Mercedes', '2025-09-24');

-- Insertar Usuarios (Users)

INSERT INTO "users\_user" ("id", "password", "is\_superuser", "username", "first\_name", "last\_name", "email", "is\_staff", "is\_active", "date\_joined", "employee\_id", "default\_off\_day", "shift\_id") VALUES

(1, 'pbkdf2\_sha256$390000$....', 0, 'jdoe', 'John', 'Doe', 'john.doe@example.com', 0, 1, '2025-01-01 10:00:00', '12345', 1, 1),

(2, 'pbkdf2\_sha256$390000$....', 0, 'asmith', 'Ana', 'Smith', 'ana.smith@example.com', 0, 1, '2025-01-01 10:00:00', '67890', NULL, 3),

(3, 'pbkdf2\_sha256$390000$....', 0, 'cgarcia', 'Carlos', 'García', 'carlos.garcia@example.com', 0, 1, '2025-01-01 10:00:00', '54321', 7, 2);

-- Insertar Registros de Trabajo (WorkLogs) - 10 registros de ejemplo

INSERT INTO "payroll\_worklog" ("id", "work\_date", "alerts\_worked", "apply\_taxi", "employee\_id", "work\_type\_id") VALUES

(1, '2025-08-02', 50, 0, 1, 1),

(2, '2025-08-03', 0, 1, 2, 5),

(3, '2025-08-04', 0, 1, 2, 5),

(4, '2025-08-09', 45, 0, 3, 2),

(5, '2025-08-16', 60, 1, 2, 4), -- Coincide con el feriado 'Día de la Restauración'

(6, '2025-08-17', 55, 0, 1, 1),

(7, '2025-08-18', 0, 1, 2, 3),

(8, '2025-08-23', 0, 1, 1, 5),

(9, '2025-08-24', 30, 0, 3, 2),

(10, '2025-08-25', 0, 0, 2, 5);

Este plan actualizado y completo integra de manera fluida la nueva funcionalidad de gestión de días feriados, manteniendo la estructura organizada y escalable del proyecto.