Universidad Mariano Gálvez de Guatemala
Ingeniería en Sistemas
Jornada Sábados
Programación III
Sección A
Ing. Melvin Cali

Proyecto 3:
Hoja de Cálculo.

Integrantes:	**Carnet**
Kevin Rai Salazar Pérez	7690-15-3698
Edwin Geovany Valle Benito	7690-19-482
Jostin Andrés Juárez González	7690-20-11493

Introducción

Este manual técnico describe el funcionamiento de una aplicación web para la gestión de hojas de cálculo, desarrollada en Python usando el framework Flask. La aplicación permite crear, modificar y gestionar hojas de cálculo de manera dinámica mediante una API RESTful.

Requisitos Previos

Para poder ejecutar esta aplicación, se necesitan los siguientes requisitos previos (Más adelante mostramos como instalarlas):

Python 3.6 o superior
Flask
Marshmallow
asteval
re (módulo estándar de Python)

Instalación y Configuración

Clonar el repositorio o copiar el código fuente.

git clone https://github.com/Jostin02/Hoja-de-Calculo.git

Instalar las dependencias necesarias (mencionadas anteriormente):
Se recomienda instalarlas en base al archivo requirements.txt

pip install -r requirements.txt

Ejecutar la aplicación:

python app.py

Esto iniciará el servidor en modo de depuración en http://127.0.0.1:5000.

Estructura del Código

Modelos

Clase Cell

Representa una celda en una hoja de cálculo.

```
class Cell:
    def __init__(self, value=''):
        self.value = value
```

Clase SpreadsheetModel

Maneja una hoja de cálculo con celdas.

```
class SpreadsheetModel:
    def __init__(self, rows, cols):
        self.data = [[Cell() for _ in range(cols)] for _ in range(rows)]
        self.asteval = Interpreter()

    def add_row(self):
        cols = len(self.data[0])
        self.data.append([Cell() for _ in range(cols)])

    def add_col(self):
        for row in self.data:
            row.append(Cell())

    def set_value(self, row, col, value):
        self.data[row][col].value = value

    def get_value(self, row, col):
        return self.data[row][col].value

    def evaluate_formula(self, formula):
        cell_ref_pattern = re.compile(r'([A-Z]+)([0-9]+)')

        def cell_ref_to_value(match):
            col_str, row_str = match.groups()
            row = int(row_str) - 1
            col = sum((ord(char) - 65 + 1) for char in col_str) - 1
            return self.get_value(row, col)

        try:
            formula_with_values = cell_ref_pattern.sub(
                lambda match: str(cell_ref_to_value(match)), formula)
            return self.asteval(formula_with_values)
        except Exception as e:
            return str(e)
```

Clase Workbook

Maneja múltiples hojas de cálculo.

```
class Workbook:
    def __init__(self):
        self.sheets = {}
        self.active_sheet = None

    def add_sheet(self, name, rows, cols):
        self.sheets[name] = SpreadsheetModel(rows, cols)
        if self.active_sheet is None:
            self.active_sheet = name

    def get_sheet(self, name):
        return self.sheets.get(name)

    def set_active_sheet(self, name):
        if name in self.sheets:
            self.active_sheet = name

    def rename_sheet(self, old_name, new_name):
        if old_name in self.sheets and new_name not in self.sheets:
            self.sheets[new_name] = self.sheets.pop(old_name)
            if self.active_sheet == old_name:
                self.active_sheet = new_name
```

Esquemas de Validación

Se utiliza Marshmallow para la validación de datos.

```
class SetValueSchema(Schema):
    sheet_name = fields.String(required=True)
    row = fields.Integer(required=True)
    col = fields.Integer(required=True)
    value = fields.String(required=True)

set_value_schema = SetValueSchema()
```

Rutas de la API

Obtener los datos de una hoja de cálculo

```
@app.route('/get_sheet', methods=['GET'])
def get_sheet():
    sheet_name = request.args.get('sheet_name', workbook.active_sheet)
    sheet = workbook.get_sheet(sheet_name)
    data = [[cell.value for cell in row] for row in sheet.data]
    return jsonify(data)
```

Obtener todas las hojas de cálculo

```
@app.route('/get_sheets', methods=['GET'])
def get_sheets():
    sheets = list(workbook.sheets.keys())
    return jsonify(sheets)
```

Establecer el valor de una celda

```
@app.route('/set_value', methods=['POST'])
def set_value():
    try:
        data = set_value_schema.load(request.json)
    except ValidationError as err:
        return jsonify(err.messages), 400

    sheet_name = data['sheet_name']
    row = data['row']
    col = data['col']
    value = data['value']
    sheet = workbook.get_sheet(sheet_name)
    if value.startswith('='):
        value = sheet.evaluate_formula(value[1:])
    sheet.set_value(row, col, value)
    return jsonify(success=True)
```

Agregar una nueva hoja de cálculo

```
@app.route('/add_sheet', methods=['POST'])
def add_sheet():
    name = request.json['name']
    rows = request.json['rows']
    cols = request.json['cols']
    workbook.add_sheet(name, rows, cols)
    return jsonify(success=True)
```

Renombrar una hoja de cálculo

```
@app.route('/rename_sheet', methods=['POST'])
def rename_sheet():
    old_name = request.json['old_name']
    new_name = request.json['new_name']
    workbook.rename_sheet(old_name, new_name)
    return jsonify(success=True)
```

Agregar una fila a una hoja de cálculo

```
@app.route('/add_row', methods=['POST'])
def add_row():
    sheet_name = request.json['sheet_name']
    sheet = workbook.get_sheet(sheet_name)
    sheet.add_row()
    return jsonify(success=True)
```

Agregar una columna a una hoja de cálculo

```
@app.route('/add_col', methods=['POST'])
def add_col():
    sheet_name = request.json['sheet_name']
    sheet = workbook.get_sheet(sheet_name)
    sheet.add_col()
    return jsonify(success=True)
```

Establecer la hoja activa

```python
@app.route('/set_active_sheet', methods=['POST'])
def set_active_sheet():
    sheet_name = request.json['sheet_name']
    workbook.set_active_sheet(sheet_name)
    return jsonify(success=True)
```

Página principal

```python
@app.route('/')
def index():
    return render_template('index.html')
```

Consideraciones de Seguridad

Validar siempre la entrada del usuario para evitar inyecciones y otros ataques.
Utilizar HTTPS para proteger los datos en tránsito.
Manejar correctamente las excepciones para no revelar detalles internos del servidor.

Conclusión

Este manual proporciona una visión general detallada del sistema de hoja de cálculo basado en Flask, incluyendo su estructura de código y rutas de la API. Siguiendo este manual, los desarrolladores pueden entender y extender el sistema según sus necesidades.