Servacal.Auxiliar - Complete Code Analysis and Documentation

Table of Contents

Project Overview

Servacal.Auxiliar is a Python-based invoice generation system designed for the Servacal_Spv company. The system automates the process of creating professional invoices by processing client data and work/service records from CSV files.

Key Features:

• Automated invoice generation from CSV data

• Flexible client and work data management

• Tax calculation with configurable rates

• Professional invoice formatting

• File export capabilities

• Robust error handling and data validation

File Structure Analysis

Core Files:

1. `main.py` (7 lines)

• **Purpose**: Entry point of the application

• **Content**: Simple main function that prints a greeting message

• **Current State**: Basic placeholder implementation

2. `invoice.py` (378 lines)

• **Purpose**: Core invoice generation system

• **Content**: Complete `InvoiceGenerator` class with all business logic

• **Key Components**:

• Data loading and validation

• Client and work data management

• Invoice calculation and formatting

• File export functionality

3. `pyproject.toml` (10 lines)

• **Purpose**: Project configuration and dependency management

• **Dependencies**: pandas>=2.3.2

• **Python Version**: >=3.11

4. `README.md` (10 lines)

• **Purpose**: Project documentation

• **Content**: Brief description of the robot created for IA course

Data Files:

5. `clients.csv` (5 lines)

• **Purpose**: Client database

• **Structure**: id, name, address, zip_code

• **Sample Data**: 3 clients (ABC Corporation, John Doe, XYZ Ltd)

6. `works.csv` (6 lines)

• **Purpose**: Work/services database

• **Structure**: client_id, date, description, amount

• **Sample Data**: 4 work records for different clients

7. `invoice_INV-20250905-12345.txt` (21 lines)

• **Purpose**: Generated invoice output

• **Content**: Formatted invoice for client 12345

Detailed Code Analysis

InvoiceGenerator Class

The `InvoiceGenerator` class is the heart of the system, containing all the business logic for invoice processing.

Constructor (`__init__`)

```
def __init__(self, clients_csv_path, works_csv_path, tax_rate=0.21):
```

• **Parameters**:

• `clients_csv_path`: Path to client data CSV file

• `works_csv_path`: Path to work/services data CSV file

• `tax_rate`: Tax rate (default 21%)

• **Functionality**: Initializes the generator and loads data from CSV files

Data Loading Methods

`load_clients_data()`

• **Purpose**: Loads client information from CSV

• **Features**:

• Error handling for missing files

• Column name cleaning (removes whitespace)

• Progress reporting

• **Returns**: pandas DataFrame with client data

`load_works_data()`

• **Purpose**: Loads work/services data from CSV

• **Features**:

• Automatic date column detection and conversion

• Column name cleaning

• Error handling

• **Returns**: pandas DataFrame with work data

Data Retrieval Methods

`get_client_data(client_id)`

• **Purpose**: Retrieves specific client information

• **Features**:

• Flexible column name matching (id, client_id, ID, etc.)

• Type conversion handling (string comparison)

• Debug information for troubleshooting

• **Returns**: Dictionary with client data or None if not found

`get_client_works(client_id)`

• **Purpose**: Retrieves all work records for a specific client

• **Features**:

• Flexible client ID column matching

• Returns filtered DataFrame

• **Returns**: pandas DataFrame with client's work records

Calculation Methods

`calculate_invoice_totals(works_df)`

• **Purpose**: Calculates invoice financial totals

• **Features**:

• Flexible amount column detection

• Subtotal calculation (sum of all amounts)

• Tax calculation (subtotal × tax_rate)

• Total calculation (subtotal + tax)

• Rounding to 2 decimal places

• **Returns**: Dictionary with subtotal, tax, and total amounts

Invoice Generation

`generate_invoice(client_id, invoice_number, invoice_date)`

• **Purpose**: Creates complete invoice data structure

• **Features**:

• Automatic invoice number generation (format: INV-YYYYMMDD-client_id)

• Automatic date setting (current date if not provided)

• Complete invoice data assembly

• **Returns**: Dictionary containing all invoice information

Output Methods

`print_invoice(invoice)`

• **Purpose**: Displays formatted invoice in console

• **Features**:

• Professional formatting with separators

• Flexible field mapping for client information

• Tabular work/service display

• Financial summary with tax breakdown

• **Output**: Console-formatted invoice

`save_invoice_to_file(invoice, filename)`

• **Purpose**: Saves invoice to text file

• **Features**:

• Automatic filename generation

• Output redirection to file

• Error handling for file operations

• **Output**: Text file with formatted invoice

Main Function and Demo

The `main()` function demonstrates the complete workflow:

• **Initialization**: Creates InvoiceGenerator instance

• **Data Setup**: Generates sample CSV files

• **Invoice Generation**: Creates invoice for client "12345"

• **Output**: Prints and saves the invoice

Sample Data Generation

The system creates sample data for demonstration:

• **Clients**: 3 sample clients with different information

• **Works**: 4 work records across different clients

• **Purpose**: Allows immediate testing without external data

Data Flow and Architecture

Input Data Flow

```
CSV Files → InvoiceGenerator → Processed Data → Invoice Output
```

• **Data Loading**: CSV files are read and loaded into pandas DataFrames

• **Data Validation**: Column names are cleaned and validated

• **Data Processing**: Client and work data are matched and processed

• **Calculation**: Financial totals are calculated

• **Formatting**: Data is formatted into professional invoice layout

6. **Output**: Invoice is displayed and/or saved to file

Data Structure Relationships

```
clients.csv (id) ←→ works.csv (client_id) ↓ ↓ Client Data Work Records ↓
↓ ■■■■■→ Invoice Generation ←■■■■■
```

Key Data Mappings

Client Data Fields

• **ID Fields**: id, client_id, ID, Client_ID, id_number, ID_Number

• **Name Fields**: name, company_name, client_name, Name, Company_Name, Client_Name

• **Address Fields**: address, Address, street, Street

• **Location Fields**: zip_code, zip, postal_code, Zip_Code, ZIP, Postal_Code

Work Data Fields

• **Client Reference**: client_id, Client_ID, id, ID, client, Client

• **Date Fields**: date, Date, work_date, Work_Date

• **Description Fields**: description, concept, service, Description, Concept, Service

• **Amount Fields**: amount, import, price, cost, value, total, Amount, Import, Price

Usage Examples

Basic Usage

```
# Initialize generator generator = InvoiceGenerator(
clients_csv_path="clients.csv", works_csv_path="works.csv", tax_rate=0.21
) # Generate invoice for client invoice =
generator.generate_invoice("12345") # Display and save
generator.print_invoice(invoice) generator.save_invoice_to_file(invoice)
```

Advanced Usage

```
# Custom invoice with specific number and date invoice =
generator.generate_invoice( client_id="12345",
```

```
invoice_number="CUSTOM-001", invoice_date="2024-12-31" )
```

Dependencies and Requirements

Required Dependencies

• **pandas>=2.3.2**: Data manipulation and CSV processing

• **Python>=3.11**: Minimum Python version

Built-in Dependencies

• **datetime**: Date and time handling

• **pathlib**: File path operations

• **sys**: System-specific parameters and functions

Optional Dependencies

• **csv**: CSV file handling (built-in)

• **io**: Input/output operations (built-in)

Error Handling and Robustness

File Operations

• **File Not Found**: Graceful handling with informative error messages

• **Permission Errors**: Exception handling for file access issues

• **Invalid Data**: Type conversion and validation

Data Validation

• **Column Detection**: Flexible column name matching

• **Type Conversion**: String conversion for ID matching

• **Missing Data**: Null value handling with pd.notna()

User Experience

• **Debug Information**: Detailed logging for troubleshooting

• **Progress Reporting**: Status messages for data loading

• **Error Messages**: Clear, actionable error descriptions

Future Improvements

Code Enhancements

• **Configuration File**: External configuration for column mappings

• **Database Support**: Direct database connectivity

• **PDF Export**: Professional PDF invoice generation

• **Email Integration**: Automatic invoice delivery

• **Web Interface**: Browser-based invoice management

Feature Additions

• **Multiple Tax Rates**: Support for different tax types

• **Currency Support**: Multi-currency invoice generation

• **Template System**: Customizable invoice templates

• **Batch Processing**: Multiple invoice generation

• **Audit Trail**: Invoice history and tracking

Performance Optimizations

• **Caching**: Data caching for improved performance

• **Async Processing**: Asynchronous file operations

• **Memory Management**: Efficient data handling for large datasets

• **Validation Optimization**: Faster data validation

Conclusion

The Servacal.Auxiliar system provides a solid foundation for automated invoice generation with its flexible data handling, robust error management, and professional output formatting. The codebase demonstrates good software engineering practices with clear separation of concerns, comprehensive error handling, and extensible architecture.

The system is particularly well-suited for small to medium businesses that need to process invoices from CSV data sources, with the flexibility to handle various data formats and the robustness to handle real-world data inconsistencies.

Generated on: 2025-01-27

System Version: Servacal.Auxiliar v0.1.0