Project Documentation

Overview

This project is a solution that integrates the pumped oil orders from the OilMat system with workshop ERP solutions in order to free the workshop manager from the work of ensuring that the corrosponding orderline is added to the customer invoice. Components:

- A flask api to handle the incomming orders.
- A worker that consumes the orders and send them to workshop ERP solution.
- The queue that handles comunication between api and worker

In addition there will be a number of changes in both the workshop and ILX management apis to allow users to configure and handle each integration. Philip will specify these changes in a seperate doc referenced from here.

Workshop integration configuration and handling:

- ERP system
- User credentials
- Active indicator
- Activate/deactivate integration

Management api:

- Get health overview for integrations
- Start/Stop integration api
- Start/Stop integration worker
- Kill integration api.

Currently its planed to start a seperate flask server/api and consumer pr workshop ERP integration. This ensures that problems conserning one workshop wont affect others and allow for easy scalability.

The application provides the below endpoints to interact with the workshop ERP system:

GET /alive

Checks if the API is alive and if the worker is running. **Response:**

- `200 OK` with a JSON object indicating the status of API + worker and the workshop that the api instance is serving.

PUT /kill

Kills an api instance. **Response:**

- `200 OK` with a JSON object indicating weather the api instance were running or not.

GET /queue

Retrieves the current tasks in the queue.

Response:

- `200 OK` with a JSON object containing the list of tasks in the queue and the length of the queue.

PUT /clear_queue

Clears all tasks from the queue.

Response:

- `200 OK` with a JSON object indicating the queue has been cleared.

PUT /start_worker

Starts the worker thread to process tasks in the queue.

Response:

- `200 OK` with a JSON object indicating the worker has started.

PUT /stop_worker

Stops the worker thread.

Response:

- `200 OK` with a JSON object indicating the worker has stopped.

POST /create

Adds a task to create an order line to the queue.

Request Body:

```
{
    "workshop": "Bennys Auto",
    "worksheet": "558",
    "product_nr": "10",
    "product_amount": "5",
    "uniqueid": "xxx",
    "username": "admin",
    "password":  "gygag",
}
```

Response:

- `200 OK` with a JSON object indicating the task has been added to the queue.
- `400 Bad Request` if any required parameters are missing.

GET /get_order_status

Get the current status of a placed order, an order will go trough the following statuses:

- placed
- processing
- created

In case of problems in the flow the returned status will be error and a reason code and a reason text is supplied

Response:

- `200 OK` with a JSON object indicating the task has been added to the queue.
- `400 Bad Request` if any required parameters are missing.

Files and Directories

```
api.py - The API server file using Flask.

WORKSHOP_api.log - Log file for the API.

ERPSYSTEM_create_orderline.py - Script for creating order lines using
Selenium. There will be as seperate file pr ERP integration.

WORKSHOP_create_orderline.log - Log file for admanager integration

api.py - Main API server file using Flask.

create_client.py - Test script for creating an orderline via the api

requirements.txt - File listing all the dependencies required to run the
application.

task_queue - Directory containing task queue files.
```

Install

Dependencies

The solution uses the following python components:

- Flask
- requests
- cryptography
- dotenv
- selenium
- persistqueue

Install the dependencies using pip:

```
pip install -r requirements.txt
```

Start Integration

Durring normal operation an ERP integration needs to be started/restarted in the following situations:

1. A new integration is configured in gui
2. An existing integration is reconfigered in gui
3. Its specificaly requested in the ILX Systems admin gui
4. The backend server is restarted. All configured active ERP integrations must be started automaticly at server startup

Shell command to start an api instances

```
python api.py port workshop
```