VTT Application Setup

Status	Inbox
Favorite	☐
Archive	☐
Created time	@November 26, 2024 6:53 PM
Last edited time	@November 28, 2024 2:52 PM
Projects	VTT MVP

Developer Tools Setup

To setup the app and run it on windows, you'll need to install:

- Visual Studio Code
- Git Bash (https://maketecheasier.com/install-git-bash-on-windows/)
- For the frontend:
    - NodeJs (nodejs.org/en/download/prebuilt-installer) version 20.18.1
- For the backend:
    - Docker Desktop
    - Miniconda (for python virtual environments management) : https://docs.anaconda.com/miniconda/install/ and register it as the primary python source
    - pgAdmin 4 (useful to edit and view postgres database content) : https://www.pgadmin.org/download/pgadmin-4-windows/
    - the Stripe CLI to test user subscriptions : https://docs.stripe.com/stripe-cli?install-method=windows

Python Virtual Environments Setup (for backend, RAG)

The VTT application is a microservices application. Some services will need python 3.12, while others will need python 3.11. Because of that, we use python virtual environments. The miniconda scripts (with the conda command) will enable us to do that.

A few steps:

- add the conda scripts path to your PATH environment variable (should be like C:\Users\myusername\miniconda3\Scripts)
- open a Terminal, then run

```
conda create -n p3_12 python=3.12
```

(answer Yes to the package list), and then activate the newly created environment and install poetry

```
conda activate p3_12
pip install poetry
```

We also create another environment using python 3.11

```
conda create -n p3_11 python=3.11
conda activate p3_11
pip install poetry
```

Cloning the Git Repositories

To access the source code, you need to clone the repositories.

- Open a windows Terminal, and create a new folder to hold the code repositories:

```
mkdir vtt-workshop
cd vtt-workshop
```

- Clone the backend, frontend and rag repositories:

```
git clone https://github.com/evkosoft/vtt-workshop-back.git
git clone https://github.com/evkosoft/vtt-workshop-front.git
git clone https://github.com/evkosoft/vtt-workshop-rag.git
```

- You should then have the 3 folders (vtt-workshop-back, vtt-workshop-front and vtt-workshop-rag) in the vtt-workshop parent folder.

Running the backend

- Open the vtt-workshop-back API in Visual Studio Code:

```
cd .\vtt-workshop-back\
code .
```

- Open a Terminal in VS Code, and type into it

```
conda activate p3_12
poetry install
```

This will install all the dependencies of the backend.

Then launch a docker container running a postgresql database:

```
docker run --name local-postgres -p 5432:5432 -e POSTGRES_PAS
SWORD=root -d postgres
```

- Then follow the instructions in the README of the project to launch the backend.
- Once running, the backend API should be available at http://127.0.0.1:8000/

Running the frontend

- Open the vtt-workshop-back API in Visual Studio Code:

```
cd .\vtt-workshop-front\
code .
```

- Make a copy of the .env.example file as .env.local and fill it with your own environment variables
- Open a Terminal in VS Code, and type into it

```
npm install
npm run dev
```

- The application should be available on http://localhost:5173/

Running the RAG as a compose stack in Docker

The RAG system has actually 2 microservices : a pg-vector database and the private-GPT forked API

- To run both of these services on Docker, simply go inside the project folder and open it in VS Code:

```
cd .\vtt-workshop-rag\
code .
```

- Make a copy of the .env.example file and rename it as .env
- In the .env file, enter the API keys for OpenAI and Leonardo

- Open a Terminal in VS Code, and then execute the docker-compose command:

```
docker-compose up -d
```

- If you change the code of the rag service and want to rebuild, stop the containers, delete the service named vtt-workshop-rag and relaunch with:

```
docker-compose up --build vtt-workshop-rag -d
```

Running the RAG locally as a FastAPI application

- Go inside the project folder and open it in VS Code:

```
cd .\vtt-workshop-rag\
code .
```

- Make a copy of the .env.example file and rename it as .env
- In the .env file, enter the API keys for OpenAI and Leonardo
- Open a Terminal in VS Code, and then execute this command:

```
conda activate p3_11
poetry install --extras "ui llms-openai embeddings-openai vec
tor-stores-postgres”
```

- We need to launch an instance of a vector DB (using docker) :

```
docker run -d --name pgvect-dev  -p 5433:5432 -e POSTGRES_DB=
postgres -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postg
res pgvector/pgvector:pg16
```

- Copy the setenv.ps1.example as setenv.ps1 and edit the empty environment variables (add your OpenAI and Leonardo API keys)

```
$Env:PGPT_PROFILES="openai”
$Env:OPENAI_API_KEY=
$Env:LEONARDO_API_KEY=
$Env:VTT_WEBHOOK_URL="http://127.0.0.1:8000/genai/image/webho
ok"
$Env:DATABASE_PORT=5433
```

- Finally, run the app locally using:

```
poetry run python -m uvicorn private_gpt.main:app --reload --
port 8001
```