helix track Core

quality gate passed

JWT COMPATIBLE

The Core module (micro-service) for the Helix Track.

Development

The helix track Core has been developed and tested on AltBase Linux distribution.

Before you start

Clone the project, then, initialize and update the Git submodules.

Note: We strongly suggest you to use the `clone` script for this. See next section.

After you have cloned the project execute the `sync` script:

```
./sync
```

Note: If your Git account does not have the push permissions given by the administrator, instead of the `sync` script do execute `pull_all`:

```
./pull_all
```

Using the `clone` script

To do this automatically execute the following:

```
(test -e ./clone || wget "https://raw.githubusercontent.com/red-elf/Project-
Toolkit/main/clone?append="$(($(date +%s%N)/1000000)) -O clone) && \
    chmod +x ./clone && ./clone git@github.com:Helix-Track/Core.git ./Core &&
cd ./Core && ./open
```

or via one of the mirror repositories:

- GitFlic:

```
(test -e ./clone || \
    wget "https://gitflic.ru/project/red-elf/project-toolkit/blob/raw?
file=clone&inline=false&append="$(($(date +%s%N)/1000000)) -O clone) && \
    chmod +x ./clone && ./clone git@gitflic.ru:helix-track/core.git ./Core &&
cd ./Core && ./open
```

- Gitee:

```
(test -e ./clone || wget "https://gitee.com/Kvetch_Godspeed_b073/Project-
Toolkit/raw/main/clone?append="$(($(date +%s%N)/1000000)) -O clone) && \
    chmod +x ./clone && ./clone git@gitee.com:Kvetch_Godspeed_b073/Core.git
./Core && cd ./Core && ./open
```

Note: It is required to execute the script from empty directory where you whish to clone the helix track project.

Executing the inititialisation scripts

Tbd.

Opening the project

From the root of the project execute:

```
./open
```

Note: The `open` command will open the project. Command will perform all the setup work for the project:

- Check if VSCode is available. If it is not, it will download it, and configure it with all mandatory development dependencies,
- Execute all prepare scripts
- Finally, will open the project.

Testing the project

From the root of the project execute:

```
./test
```

It will execute all the Testable system components.

Propriatery submodules

For the development purposes it is possible to work with the propriatery modules. To clone and link propriatery submodule do the following steps:

- Export the following environment variables:

```
export SUBMODULES_PRIVATE_HOME=/path/to/propriatery/modules/home
export SUBMODULES_PRIVATE_RECIPES=/path/to/propriatery/modules/recipes
```

- Add the recipe(s) into the reipes directory. For example `Some_Propriatery_Module.submodule`:

```
#!/bin/bash

NAME="Some_Propriatery_Module"
REPO="git@github.com:Something/Some_Propriatery_Module.git"
```

This will instruct the `open` script to clone and link the propriatery module with the project.

The cloned propriatery module will be linked under the `_Private` directory of the project under the modules name: `_Private\Some_Propriatery_Module`.

Database

The system database

The `Definition.sqlite` represents the system database. It contains all the tables and initial data required for the system to work.

The DDL directory contains all major SQL scripts required to initialize the database.

Convention used for the SQl script is the following:

- The main version scripts:

`Definition.VX.sql` where X represents the version of the database (1, 2, 3, etc).

- Migration scripts:

`Migration.VX.Y.sql` where X represents the version of the database (1, 2, 3, etc) and Y the version of the patch (1, 2, 3, etc).

All SQL scripts are executed by the shell and the `Definition.sqlite` is created as a result.

Scripts and tools

Tbd.

All scripts and tools required for the system to initialize (database, generated code, etc.)

Tbd.

Developers documentation

Documentation can be found here.