Template Application

Manual

unknown

Apr 24, 2024

Table of Contents

Chapter 1

General Documentation

This section contains the core documentation for setting up and starting with IO-TEMPLATE-APP. It covers everything from installation to basic and advanced configurations.

1.1 Introduction

TODO

1.2 Requirements

The required software is listed below. Regarding the corresponding software versions, you will find the detailed information in the Release Notes.

1.2.1 Operating System

Continuous delivery / integration (CD/CI) runs on Ubuntu and development is also done with macOS and Windows 10/11.

The installation of Homebrew is required for macOS. If necessary, Homebrew can be installed with the following command:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

For the Windows operating systems, only additional the functionality of the `make` tool must be made available, e.g. via Make for Windows

The command-line shells supported are:

Operating system	Command-line shell(s)
macOS	zsh
Ubuntu	bash
Windows 10/11	cmd and PowerShell

For macOS and Ubuntu, the end-of-line character and the execution authorization may need to be adjusted for the shell scripts. If the `dos2Unix` program is installed, the necessary adjustments can be made using the scripts `./scripts/run_prep_zsh_scripts.zsh` (macOS) or `./scripts/run_prep_bash_scripts.sh` (Ubuntu).

1.2.2 Python

This project utilizes Python from version 3.10, which introduced significant enhancements in type hinting and type annotations. These improvements provide a more robust and clear definition of

function parameters, return types, and variable types, contributing to improved code readability and maintainability. The use of Python 3.12 ensures compatibility with these advanced typing features, offering a more structured and error-resistant development environment.

1.2.3 Docker Desktop

The project employs PostgreSQL for data storage and leverages Docker images provided by PostgreSQL to simplify the installation process. Docker Desktop is used for its ease of managing and running containerized applications, allowing for a consistent and isolated environment for PostgreSQL. This approach streamlines the setup, ensuring that the database environment is quickly replicable and maintainable across different development setups.

1.2.4 Miniconda

Some of the Python libraries required by the project are exclusively available through Conda. To maintain a minimal installation footprint, it is recommended to install Miniconda, a smaller, more lightweight version of Anaconda that includes only Conda, its dependencies, and Python.

By using Miniconda, users can access the extensive repositories of Conda packages while keeping their environment lean and manageable. To install Miniconda, follow the instructions provided in the `scripts` directory of the project, where operating system-specific installation scripts named `run_install_miniconda` are available for Windows (CMD shell), Ubuntu (Bash shell), and macOS (Zsh shell).

Utilizing Miniconda ensures that you have the necessary Conda environment with the minimal set of dependencies required to run and develop the project efficiently.

1.2.5 MS Access Database Engine

This Software consists of a set of components that facilitate the transfer of data between existing Microsoft Office files such as Microsoft Office Access (*.mdb and *.accdb) files and Microsoft Office Excel (*.xls, *.xlsx, and *.xlsb) files to other data sources. Connectivity to existing text files is also supported.

1.2.6 DBeaver Community - optional

DBeaver is recommended as the user interface for interacting with the PostgreSQL database due to its comprehensive and user-friendly features. It provides a flexible and intuitive platform for database management, supporting a wide range of database functionalities including SQL scripting, data visualization, and import/export capabilities. Additionally, the project includes predefined connection configurations for DBeaver, facilitating a hassle-free and streamlined setup process for users.

1.3 Installation

1.3.1 Python

The project repository contains a `scripts` directory that includes operating system-specific installation scripts for Python, ensuring a smooth setup across various environments.

- **macOS**: The `run_install_python.zsh` script is available for macOS users. This script is adapted for the Zsh shell, which is the standard shell on recent versions of macOS, and it streamlines the Python installation.
- **Ubuntu**: For users on Ubuntu, the `run_install_python.sh` script is provided. This Bash script is created to operate within the default shell environment of Ubuntu, facilitating the Python installation process.
- **Windows**: The `run_install_python.bat` script is tailored for users on Windows systems. It is designed to be run in the Command Prompt and automates the Python installation process on

Windows.

These scripts are named according to the convention `run_install_python.<ext>`, where `<ext>` corresponds to the script extension appropriate for the target operating system and shell environment (e.g., *.bat* for Windows, *.sh* for Ubuntu Bash, *.zsh* for macOS). Users are recommended to execute the script matching their OS to ensure an efficient Python setup.

1.3.2 AWS Command Line Interface

Within the project's `scripts` directory, you will find a set of scripts specifically designed for the installation of the AWS Command Line Interface (AWS CLI). These scripts facilitate the installation process on different operating systems, ensuring a consistent and reliable setup.

- **macOS**: For macOS users, the `run_install_aws_cli.zsh` script is provided. Designed for the Zsh shell, this script streamlines the AWS CLI installation process on macOS by leveraging the Homebrew package manager.
- **Ubuntu**: Ubuntu users should utilize the `run_install_aws_cli.sh` script. This script is a Bash script that simplifies the AWS CLI installation on Ubuntu systems by setting up the necessary repositories and installing the CLI via apt-get.
- **Windows**: The `run_install_aws_cli.bat` script is intended for Windows users. It automates the process of downloading and installing the latest version of the AWS CLI in the Windows Command Prompt environment.

Each script is named following the pattern `run_install_aws_cli.<ext>`, with `<ext>` being the respective script extension suitable for the target operating system and shell environment (e.g., *.bat* for Windows CMD, *.sh* for Ubuntu Bash, *.zsh* for macOS Zsh). Users are advised to execute the corresponding script for their operating system to achieve an optimal AWS CLI installation experience.

1.3.3 Miniconda

The `scripts` directory includes a collection of operating system-specific scripts named `run_install_miniconda` to streamline the installation of Miniconda. These scripts are designed to cater to the needs of different environments, making the setup process efficient and user-friendly.

- **Windows CMD Shell**: The `run_install_miniconda.bat` script is tailored for the Windows CMD shell. It automates the Miniconda installation process on Windows, providing a hassle-free setup with a simple double-click or command line execution.
- **Ubuntu Bash Shell**: Ubuntu users can take advantage of the `run_install_miniconda.sh` script. This Bash script is intended for use within the Ubuntu terminal, encapsulating the necessary commands to install Miniconda seamlessly on Ubuntu systems.
- **macOS Zsh Shell**: For macOS, the `run_install_miniconda.zsh` script is available. It is optimized for the Zsh shell, which is the default on recent versions of macOS. This script simplifies the installation and configuration of Miniconda, ensuring a smooth integration with macOS.

Each script in the `scripts` directory is named to reflect its compatibility with the corresponding operating system and shell environment. Users are encouraged to execute the script that matches their OS for a smooth and error-free Miniconda installation experience.

1.3.4 Docker Desktop

The `scripts` directory contains scripts that assist with installing Docker Desktop on macOS and Ubuntu, facilitating an automated and streamlined setup.

- **macOS**: The `run_install_docker.zsh` script is designed for macOS users. By utilizing this Zsh script, the installation of Docker Desktop on macOS is executed through a series of automated steps, which are managed by the script to ensure a smooth installation process.
- **Ubuntu**: The `run_install_docker.sh` script is available for Ubuntu users. This Bash script

sets up Docker Desktop on Ubuntu systems by configuring the necessary repositories and managing the installation steps through the system's package manager.

- **Windows**: For Windows users, it is recommended to download and install Docker Desktop using the traditional installer available at Docker Desktop for Windows. This approach guarantees the most stable version and is tailored to integrate seamlessly with Windows-specific features and configurations.

Please select and execute the appropriate script for your operating system from the `scripts` directory. Windows users should follow the provided link to obtain the official installer for a guided installation experience.

1.3.5 MS Access Database Engine

- **Windows**: The software can be downloaded from here and then installed according to the instructions provided.
- **Ubuntu Bash Shell**: The necessary software can be downloaded with the package manager `apt` as follows:

```
sudo apt-get update -y
sudo apt-get install -y unixodbc-dev
```

- **macOS Zsh Shell**: The necessary software can be downloaded with the package manager `Homebrew` as follows:

```
brew update
brew install unixodbc
```

1.3.6 DBeaver - optional

DBeaver is an optional but highly recommended tool for this software as it offers a user-friendly interface to gain insights into the database internals. The project provides convenient scripts for installing DBeaver on macOS and Ubuntu.

- **macOS**: The `run_install_dbeaver.zsh` script is crafted for macOS systems. By running this Zsh script, users can easily install DBeaver and quickly connect to the database for management and querying tasks.
- **Ubuntu**: For Ubuntu users, the `run_install_dbeaver.sh` script facilitates the installation of DBeaver. This Bash script automates the setup process, adding necessary repositories and handling the installation seamlessly.
- **Windows**: Windows users are advised to download and install DBeaver using the official installer from the DBeaver website at DBeaver Download. The installer ensures that DBeaver is properly configured and optimized for Windows environments.

To install DBeaver, locate the appropriate script in the `scripts` directory for macOS or Ubuntu. If you're a Windows user, please use the provided link to access the official installer for an intuitive installation experience.

1.3.7 Python Libraries

The project's Python dependencies are managed partly through Conda and partly through pip. To facilitate a straightforward installation process, a Makefile is provided at the root of the project.

- **Development Environment**: Run the command `make conda-dev` from the terminal to set up a development environment. This will install the necessary Python libraries using Conda and pip as specified for development purposes.
- **Production Environment**: Execute the command `make conda-prod` for preparing a produc-

tionenvironment. It ensures that all the required dependencies are installed following the configurations optimized for production deployment.

The Makefile targets abstract away the complexity of managing multiple package managers and streamline the environment setup. It is crucial to have both Conda and the appropriate pip tool available in your system's PATH to utilize the Makefile commands successfully.

1.4 Configuration IO-TEMPLATE-APP

1.4.1 .act_secrets

This file controls the secrets of the `make action` functionality. This file is not included in the repository. The file `.act_secrets_template` can be used as a template.

The customisable entries are:

Parameter	Description
GLOBAL_USER_EMAIL	The global email address for GitHub

Examples:

```
GLOBAL_USER_EMAIL=a@b.com
```

1.4.2 .settings.io_aero.toml

This file controls the secrets of the application. This file is not included in the repository. The file `.settings.io_aero_template.toml` can be used as a template.

The customisable entries are:

Parameter	Description
postgres_password	Password of the database user
postgres_password_admin	Password of the database administrator

The secrets can be set differently for the individual environments (`default` and `test`).

Examples:

```
[default]
postgres_password = "..."
postgres_password_admin = "..."

[test]
postgres_password = "postgres_password"
postgres_password_admin = "postgres_password_admin"
```

1.4.3 settings.io_aero.toml

This file controls the behaviour of the application.

The customisable entries are:

Parameter	Description
check_value	`default` for productive operation, `test` for test operation
is_verbose	Display progress messages for processing

The configuration parameters can be set differently for the individual environments (`default` and `test`).

Examples:

```
[default]
check_value = "default"
is_verbose = true

[test]
check_value = "test"
```

1.5 Configuration Logging

In **IO-TEMPLATE-APP** the Python standard module for logging is used - details can be found here.

The file `logging_cfg.yaml` controls the logging behaviour of the application.

Default content:

```
version: 1

disable_existing_loggers: False

formatters:
  simple:
    format: "%(asctime)s [%(name)s] [%(module)s.py  ] %(levelname)-5s
%(funcName)s:%(lineno)d %(message)s"
  extended:
    format: "%(asctime)s [%(name)s] [%(module)s.py  ] %(levelname)-5s
%(funcName)s:%(lineno)d \n%(message)s"

handlers:
  console:
    class: logging.StreamHandler
    level: INFO
    formatter: simple
  file_handler:
    class: logging.FileHandler
    level: INFO
    filename: logging_io_aero.log
    formatter: extended

root:
  level: DEBUG
  handlers: [ console, file_handler ]
```

1.6 First Steps

To get started, you'll first need to clone the repository, which contains essential scripts for various operating systems. After cloning, you will use these scripts to install the necessary foundational software. Finally, you will complete the repository-specific installation to set up your environment correctly. Detailed instructions for each of these steps are provided below.

1.6.1 Cloning the Repository

Start by cloning the *io-template-app* repository. This repository contains essential scripts and configurations needed for the project.

```
git clone https://github.com/io-aero/io-template-app
```

1.6.2 Install Foundational Software

Once you have successfully cloned the repository, navigate to the cloned directory. Within the *scripts* folder, you will find scripts tailored for various operating systems. Proceed with the subsection that corresponds to your operating system for further instructions.

macOS

To set up the project on a macOS system, the following steps should be performed in a terminal window within the repository directory:

a. Grant Execute Permission to Installation Scripts

Provide execute permissions to the installation scripts:

```
chmod +x scripts/*.zsh
```

b. Install Python and pip

Run the script to install Python and pip:

```
./scripts/run_install_python.zsh
```

c. Install AWS Command Line Interface

Execute the script to install the AWS CLI:

```
./scripts/run_install_aws_cli.zsh
```

d. Install Miniconda and the Correct Python Version

Use the following script to install Miniconda and set the right Python version:

```
./scripts/run_install_miniconda.zsh
```

e. Install Docker Desktop

To install Docker Desktop, run:

```
./scripts/run_install_docker.zsh
```

f. Install Terraform

To install Docker Desktop, run:

```
./scripts/run_install_terraform.zsh
```

g. Optionally Install DBeaver

If needed, install DBeaver using the following script:

```
./scripts/run_install_dbeaver.zsh
```

h. Close the Terminal Window

Once all installations are complete, close the terminal window.

Ubuntu

To set up the project on an Ubuntu system, the following steps should be performed in a terminal window within the repository directory:

a. Grant Execute Permission to Installation Scripts

Provide execute permissions to the installation scripts:

```
chmod +x scripts/*.sh
```

b. Install Python and pip

Run the script to install Python and pip:

```
./scripts/run_install_python.sh
```

c. Install AWS Command Line Interface

Execute the script to install the AWS CLI:

```
./scripts/run_install_aws_cli.sh
```

d. Install Miniconda and the Correct Python Version

Use the following script to install Miniconda and set the right Python version:

```
./scripts/run_install_miniconda.sh
```

e. Install Docker Desktop

This step is not required for WSL (Windows Subsystem for Linux) if Decker Desktop is installed in Windows and this is configured for WSL 2 based engine.

To install Docker Desktop, run:

```
./scripts/run_install_docker.sh
```

f. Install Terraform

To install Docker Desktop, run:

```
./scripts/run_install_terraform.sh
```

g. Optionally Install DBeaver

If needed, install DBeaver using the following script:

```
./scripts/run_install_dbeaver.sh
```

h. Close the Terminal Window

Once all installations are complete, close the terminal window.

Windows 10/11

To set up the project on a Windows 10/11 system, the following steps should be performed in a command prompt (cmd) within the repository directory:

a. Install Python and pip

Run the script to install Python and pip:

```
scripts/run_install_python.bat
```

b. Install AWS Command Line Interface

Execute the script to install the AWS CLI:

```
scripts/run_install_aws_cli.bat
```

c. Install Miniconda and the Correct Python Version

Use the following script to install Miniconda and set the right Python version:

```
scripts/run_install_miniconda.bat
```

d. Close the Command Prompt

Once all installations are complete, close the command prompt.

e. Install Docker Desktop

To install Docker Desktop, download the software from here:

https://www.docker.com/products/docker-desktop/

and follow the installation instructions.

f. Install Terraform

To install Terraform, download the software from here:

https://developer.hashicorp.com/terraform/install?product_intent=terraform

and follow the installation instructions.

g. Optionally Install DBeaver

If needed, install DBeaver, download the software from here:

https://dbeaver.io/

and follow the installation instructions.

1.6.3 Repository-Specific Installation

After installing the basic software, you need to perform installation steps specific to the *io-template-app* repository. This involves setting up project-specific dependencies and environment configurations. To perform the repository-specific installation, the following steps should be performed in a command prompt or a terminal window (depending on the operating system) the repository directory.

Setting Up the Python Environment

To begin, you'll need to set up the Python environment using Miniconda, which is already pre-installed. You can use the provided Makefile for managing the environment.

*a. For **production** use, run the following command:*

```
make conda-prod
```

*b. For **software development**, use the following command:*

```
make conda-dev
```

These commands will create and configure a virtual environment for your Python project, ensuring a clean and reproducible development or production environment. The virtual environment is automatically activated by the Makefile, so you don't need to activate it manually.

Minor Adjustments for GDAL

The installation of the GDAL library requires the following minor operating system-specific adjustments:

a. macOS

In macOS, the GDAL library must be installed as follows:

```
brew install gdal
```

b. Ubuntu

In Ubuntu, the GDAL library must be installed as follows:

```
sudo apt-get install gdal-bin libgdal-dev
```

c. Windows 10/11

Assuming that Miniconda is installed in the following file directory

```
C:\ProgramData\miniconda3
```

then the following entry must then be added to the path variable:;

```
C:\ProgramData\miniconda3\envs\iotemplateapp\Library\bin
```

System Testing with Unit Tests

If you have previously executed *make conda-dev*, you can now perform a system test to verify the installation using *make test*. Follow these steps:

a. Run the System Test:

Execute the system test using the following command:

```
make tests
```

This command will initiate the system tests using the previously installed components to verify the correctness of your installation.

b. Review the Test Results:

After the tests are completed, review the test results in the terminal. Ensure that all tests pass without errors.

If any tests fail, review the error messages to identify and resolve any issues with your installation.

Running system tests using *make tests* is a valuable step to ensure that your installation is working correctly, and your environment is properly configured for your project. It helps identify and address any potential problems early in the development process.

Downloading Database Files (Optional)

Database files can be downloaded from the IO-Aero Google Drive directory *io_aero_data/TO DO/database/TO DO* to your local repository directory *data*. Before extracting, if a *postgres* directory exists within the *data* directory, it should be deleted.

Follow these steps to manage the database files:

a. Access the IO-Aero Google Drive Directory:

Navigate to the IO-Aero Google Drive and locate the directory *io_aero_data/TO DO/database/TO DO*.

b. Download Database Files:

Download the necessary database files from the specified directory to your local repository directory *data*.

c. Delete Existing postgres Directory (if present):

If a directory named *postgres* already exists within the *data* directory, you should delete it to avoid conflicts.

d. Extract Database Files:

The downloaded database files are in an archive format (ZIP) and should be extracted in the *data* directory. After completing these steps, the database files should reside in the *data* directory of your local repository and will be ready for use.

Creating the Docker Container with PostgreSQL DB

To create the Docker container with PostgreSQL database software, you can use the provided *run_io_template_app* script. Depending on your operating system, follow the relevant instructions below:

a. macOS (zsh):

```
./scripts/run_io_template_app.zsh s_d_c
```

b. Ubuntu (sh):

```
./scripts/run_io_template_app.sh s_d_c
```

c. Windows 10/11 (cmd):

```
scripts\run_io_template_app.cmd s_d_c
```

These commands will initiate the process of creating the Docker container with PostgreSQL database software.

1.7 Advanced Usage

TODO

Chapter 2

API Documentation

Here, you will find detailed API documentation, which includes information about all modules within the IO-TEMPLATE-APP, allowing developers to understand the functionalities available.

2.1 iotemplateapp

2.1.1 iotemplateapp package

Submodules

iotemplateapp.glob_local module

Global constants and variables.

iotemplateapp.glob_local.**ARG_TASK = 'task'**
A constant key used to reference the 'task' argument in function calls and command line arguments throughout the software.

Type str

iotemplateapp.glob_local.**ARG_TASK_CHOICE = ''**
Initially set to an empty string, this variable is intended to hold the user's choice of task once determined at runtime.

Type str

iotemplateapp.glob_local.**ARG_TASK_VERSION = 'version'**
A constant key used to reference the 'version' argument for tasks, indicating the version of the task being used.

Type str

iotemplateapp.glob_local.**CHECK_VALUE_TEST = True**
A boolean indicating whether the check value from io_config is 'test'.

Type bool

iotemplateapp.glob_local.**FATAL_00_926 = "FATAL.00.926 The task '{task}' is invalid"**
Error message template indicating that the specified task is invalid.

Type str

iotemplateapp.glob_local.**INFO_00_004 = 'INFO.00.004 Start Launcher'**
Information message indicating the start of the launcher.

Type str

iotemplateapp.glob_local.**INFO_00_005 = "INFO.00.005 Argument '{task}'='{value_task}"'**
Information message indicating the value of a specific argument in the launcher.

Type str

iotemplateapp.glob_local.**INFO_00_006 = 'INFO.00.006 End Launcher'**
Information message indicating the end of the launcher.

Type str

iotemplateapp.glob_local.**INFORMATION_NOT_YET_AVAILABLE = 'n/a'**
Placeholder indicating that information is not yet available.

Type str

iotemplateapp.glob_local.**IO_TEMPLATE_APP_VERSION = '9.9.9'**
The current version number of the IO-Aero template application.

Type str

iotemplateapp.glob_local.**LOCALE = 'en_US.UTF-8'**
Default locale setting for the system to 'en_US.UTF-8', ensuring consistent language and regional format settings.

Type str

iotemplateapp.templateapp module

IO-TEMPLATE-APP interface.

iotemplateapp.templateapp.**ARG_TASK = ''**
Placeholder for the command line argument 'task'.

Type str

iotemplateapp.templateapp.**check_arg_task** (*args: Namespace*) → None
Check the command line argument: -t / –task.
Args:

args (argparse.Namespace): Command line arguments.

iotemplateapp.templateapp.**get_args** ( ) → None
Load the command line arguments into the memory.

iotemplateapp.templateapp.**progress_msg** (*msg: str*) → None
Create a progress message.
Args:

msg (str): Progress message

iotemplateapp.templateapp.**progress_msg_time_elapsed** (*duration: int, event: str*) → None
Create a time elapsed message.
Args:

duration (int): Time elapsed in ns. event (str): Event description.

iotemplateapp.templateapp.**terminate_fatal** (*error_msg: str*) → None
Terminate the application immediately.
Args:

error_msg (str): Error message

iotemplateapp.templateapp.**version**( ) → str

Return the version number of the IO-XPA-DATA application.

Returns **str**

Return type The version number of the IO-XPA-DATA application

Module contents

IO-TEMPLATE-APP.

Chapter 3

About

This section provides additional context and legal information about IO-TEMPLATE-APP, including release notes and licensing details.

3.1 Release Notes

3.1.1 Version 1.0.0

Release Date: dd.mm.2024

New Features

- TODO

Modified Features

- TODO

Deleted Features

- TODO

Applied Software

Software	Version	Remark	Status
DBeaver - optional	23.3.1		
Docker Desktop	4.26.1		
Miniconda	23.3.1		
Python	3.12.3		

Windows-specific Software

Important: All software components should be installed in the 64 bit version!

Software	Version	Remark	Status
Make for Windows	3.81		

3.2 End-User License Agreement

3.2.1 End-User License Agreement (EULA) of IO-Aero Software

This End-User License Agreement ("**EULA**") is a legal agreement between you and **IO-Aero**.

This **EULA** agreement governs your acquisition and use of our **IO-Aero Software** ("Software") directly from **IO-Aero** or indirectly through a **IO-Aero** authorized reseller or distributor (a "Reseller").

Please read this **EULA** agreement carefully before completing the installation process and using the **IO-Aero Software**. It provides a license to use the **IO-Aero Software** and contains warranty information and liability disclaimers.

If you register for a free trial of the **IO-Aero Software**, this **EULA** agreement will also govern that trial. By clicking "accept" or installing and/or using the **IO-Aero Software**, you are confirming your acceptance of the Software and agreeing to become bound by the terms of this **EULA** agreement.

If you are entering into this **EULA** agreement on behalf of a company or other legal entity, you represent that you have the authority to bind such entity and its affiliates to these terms and conditions. If you do not have such authority or if you do not agree with the terms and conditions of this **EULA** agreement, do not install or use the Software, and you must not accept this **EULA** agreement.

This **EULA** agreement shall apply only to the Software supplied by **IO-Aero** herewith regardless of whether other software is referred to or described herein. The terms also apply to any **IO-Aero** updates, supplements, Internet-based services, and support services for the Software, unless other terms accompany those items on delivery. If so, those terms apply.

License Grant

IO-Aero hereby grants you a personal, non-transferable, non-exclusive licence to use the **IO-Aero Software** on your devices in accordance with the terms of this **EULA** agreement.

You are permitted to load the **IO-Aero Software** (for example a PC, laptop, mobile or tablet) under your control. You are responsible for ensuring your device meets the minimum requirements of the **IO-Aero Software**.

You are not permitted to:

- Edit, alter, modify, adapt, translate or otherwise change the whole or any part of the Software nor permit the whole or any part of the Software to be combined with or become incorporated in any other software, nor decompile, disassemble or reverse engineer the Software or attempt to do any such things
- Reproduce, copy, distribute, resell or otherwise use the Software for any commercial purpose
- Allow any third party to use the Software on behalf of or for the benefit of any third party
- Use the Software in any way which breaches any applicable local, national or international law
- use the Software for any purpose that **IO-Aero** considers is a breach of this **EULA** agreement Intellectual Property and Ownership

IO-Aero shall at all times retain ownership of the Software as originally downloaded by you and all subsequent downloads of the Software by you. The Software (and the copyright, and other intellectual property rights of whatever nature in the Software, including any modifications made thereto) are and shall remain the property of **IO-Aero**.

IO-Aero reserves the right to grant licences to use the Software to third parties.

Termination

This **EULA** agreement is effective from the date you first use the Software and shall continue until

terminated. You may terminate it at any time upon written notice to **IO-Aero**.

It will also terminate immediately if you fail to comply with any term of this **EULA** agreement. Upon such termination, the licenses granted by this **EULA** agreement will immediately terminate, and you agree to stop all access and use of the Software. The provisions that by their nature continue and survive will survive any termination of this **EULA** agreement.

Governing Law

This **EULA** agreement, and any dispute arising out of or in connection with this **EULA** agreement, shall be governed by and construed in accordance with the laws of the United States.

Indices and tables

- genindex
- modindex

4.1 Repository

Link to the repository for accessing the source code and contributing to the project:
IO-TEMPLATE-APP GitHub Repository

4.2 Version

This documentation is for IO-TEMPLATE-APP version .

i