**ALL THE DEPENDENCIES DETAILS**

* **FastAPI**

### License: Free software: MIT license

### Requires: Python 3.6 or above

### What is FastAPI?

### FastAPI is a modern, fast, web framework for building APIs with Python 3.6+ based on standard Python type hints.

### It can handle both synchronous and asynchronous requests and has built-in support for data validation, JSON serialization, authentication and authorization, and OpenAPI (version 3.0).

### Dependency Injection

FastAPI includes an extremely easy to use, but extremely powerful **Dependency Injection** system.

* Even dependencies can have dependencies, creating a hierarchy or **"graph" of dependencies**.
* All **automatically handled** by the framework.
* All the dependencies can require data from requests and **augment the path operation** constraints and automatic documentation.
* **Automatic validation** even for *path operation* parameters defined in dependencies.
* Support for complex user authentication systems, **database connections**, etc.
* **No compromise** with databases, frontends, etc. But easy integration with all of them.

**Install using pip**

**Pip install FastAPI**

**Community’s/Support:**

1. Gitter Chat(<https://gitter.im/fast_api/community>)
2. FastAPI Discussions(<https://github.com/tiangolo/fastapi/discussions/2475>)

* **Uvicorn**

### License: BSD License (BSD)

* Uvicorn is a lightning-fast ASGI server implementation, using [uvloop](https://github.com/MagicStack/uvloop) and [httptools](https://github.com/MagicStack/httptools).
* Until recently Python has lacked a minimal low-level server/application interface for asyncio frameworks. The [ASGI specification](https://asgi.readthedocs.io/en/latest/) fills this gap, and means we're now able to start building a common set of tooling usable across all asyncio frameworks.
* Uvicorn currently supports HTTP/1.1 and WebSockets. Support for HTTP/2 is planned.

**Install using pip:**

**Pip install uvicorn**

This will install uvicorn with minimal (pure Python) dependencies.

**Pip install uvicorn [standard]**

This will install uvicorn with "Cython-based" dependencies (where possible) and other "optional extras".

In this context, "Cython-based" means the following:

* The event loop uvloop will be installed and used if possible.
* The http protocol will be handled by httptools if possible.

Moreover, "optional extras" means that:

* The websocket protocol will be handled by websockets (should you want to use wsproto you'd need to install it manually) if possible.
* The --reloader flag in development mode will use watchgod.
* Windows users will have colorama installed for the coloured logs.
* Python-dotenv will be installed should you want to use the --env-file option.
* PyYAML will be installed to allow you to provide a .yaml file to --log-config, if desired.
* **SQLAlchemy**

### License: MIT license

* SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
* It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

**Features:**

* Extremely easy to use for all the basic tasks, such as: accessing pooled connections, constructing SQL from Python expressions, finding object instances, and committing object modifications back to the database.
* Powerful enough for complicated tasks, such as: eager load a graph of objects and their dependencies via joins; map recursive adjacency structures automatically; map objects to not just tables but to any arbitrary join or select statement; combine multiple tables together to load whole sets of otherwise unrelated objects from a single result set; commit entire graphs of object changes in one step.
* Built to conform to what DBAs demand, including the ability to swap out generated SQL with hand-optimized statements, full usage of bind parameters for all literal values, fully transactionalized and consistent updates using Unit of Work.
* Modular. Different parts of SQLAlchemy can be used independently of the rest, including the connection pool, SQL construction, and ORM. SQLAlchemy is constructed in an open style that allows plenty of customization, with an architecture that supports custom data types, custom SQL extensions, and ORM plugins which can augment or extend mapping functionality.

**Community’s/Support :**

1. Gitter Room
2. IRC Channel-Freenode

**Install using pip:**

**Pip install SQLAlchemy**

This command will download the latest **released** version of SQLAlchemy from the [Python Cheese Shop](http://pypi.python.org/pypi/SQLAlchemy) and install it to your system.

* **MySQL-Connector-Python**

**License:**

Free Software under the terms of the GNU General Public License, version 2.0, as published by the Free Software Foundation.

* MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249).
* It is written in pure Python and does not have any dependencies except for the Python Standard Library.

**Install using pip:**

**Pip install mysql-connector-python**

**Community/Support:**

1. MySQL IRC channel on [Freenode](http://www.freenode.net/)(https://freenode.net/)
2. MySQL Developer Zone
3. MySQL Forums(<https://forums.mysql.com/>)

* **Bcrypt**

**License: Apache Software License (Apache License, Version 2.0)**

* The **bcrypt** is a password hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher.
* The **bcrypt** function is the default password hash algorithm for OpenBSD.
* There are implementations of **bcrypt** for C, C++, C#, Java, JavaScript, PHP, **Python** and other languages.

**Use and Features:**

The **bcrypt** hashing function allows us to build a password security platform that scales with computation power and always hashes every password with a salt.

## Encryption

* Encryption is the process of encoding a message or information in such a way that only authorized people can read it with a corresponding key and those who are not authorized cannot. The intended information or message, referred to as plaintext, is encrypted using an encryption algorithm – a cipher – generating ciphertext that can be read only if decrypted. Encryption is a two-way function. When we encrypt something, we're doing so with the intention of decrypting it later. Encryption is used to protect data when transmitted; e.g. in a mail communication.

## Hashing

* Hashing is the process of using an algorithm to map data of any size to a fixed length. This is called a hash value. Whereas encryption is a two-way function, hashing is a one-way function. While it's technically possible to reverse-hash a value, the computing power required makes it unfeasible. While encryption is meant to protect data in transmit, hashing is meant to verify that data hasn't been altered and it is authentic.
* Passwords are not stored as plaintext in databases but in hashed values.

## Salt

* Salt is a fixed-length cryptographically-strong random value that is added to the input of hash functions to create unique hashes for every input. A salt is added to make a password hash output unique even for users adopting common passwords.

## The bcrypt hashing function

* The bcrypt is a password hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher. The bcrypt function is the default password hash algorithm for OpenBSD. There are implementations of bcrypt for C, C++, C#, Java, JavaScript, PHP, Python and other languages.
* The bcrypt algorithm creates hash and salt the password for us using strong cryptography. The computation cost of the algorithm is parameterized, so it can be increased as computers get faster. The computation cost is called work factor or cost factor. It slows down the hashing, making brute force attempts harder and slower. The optimal cost factor changes over time as computers get faster. The downside of a high cost factor is increased load on system resources and affecting user experience.

**Install using pip:**

**Pip install bcrypt**

* **Pytest**

**License: MIT License**

**What is Pytest?**

**Pytest** is a framework that makes building simple and scalable tests easy. Tests are expressive and readable—no boilerplate code required. Get started in minutes with a small unit test or complex functional test for your application or library.

**Features:**

* Easy to start with and simple syntax.
* Open Source.
* Build-in support for test discovery.
* Command-line support.
* Extensibility: Plug-ins, hooks.
* Fixtures.
* Works with built-in unit tests.

**Install using pip:**

**Pip install pytest**

**Community/Support:**

[Issue Tracker](https://github.com/pytest-dev/pytest/issues)(<https://github.com/pytest-dev/pytest/issues>)

* **PyJWT**

**License:** **MIT License**

**PyJWT** is a Python library which allows you to encode and decode JSON Web Tokens (JWT). JWT is an open, industry-standard (RFC 7519) for representing claims securely between two parties.

**Install using pip:**

pip install uvicorn

**Community/Support:**

1. Stack overflow
2. pyup(<https://pyup.io/)>

* **Virtual Environment**

**License:** **MIT License**

**Requires: Python 3 or Above**

**What is Virtual Environment?**

* **Virtualenv** is a tool to create isolated Python environments.
* **Virtualenv** creates a folder which contains the entire necessary executable to use the packages that a Python project would need. It can be used standalone, in place of Pipenv.

**What is the use of Virtualenv in Python?**

At its core, the main **purpose** of **Python** virtual environments is to create an isolated environment for **Python** projects. This means that each project can have its own dependencies, regardless of what dependencies every other project has.

**Install using pip:**

Pip install virtualenv

**Community/Support:**

1. Stack overflow
2. DEV Community