Python API Development

By Sanjeev Thiyagarajan

Introduction

* This course is all about building out your very own API in Python.
* However keep in mind this course is so much more than just building out a simple API.
* First of all we are going to build out a fully featured API that include
  + Authentication
  + CRUD Operation
  + Schema Validation
  + Documentation
* However learning doesn’t stop there this course is going to extend well past just basic API development.
* We’ll also learn all of the tooling that surrounds building a complete and robust API.
* A large section of this course is dedicated to learning SQL.
* Author: I have noticed that a lot of API and Wed Development courses just quickly gloss over the SQL without diving into the nitty-gritty of how SQL databases works.
* For this course we are going to cover SQL extensively right from the basic. No previous experience needed.
* But by the time you complete this course you will be very proficient at
  + Generating database schemas
  + Core SQL concepts like
  + Primary Key, Foreign Key, Table constraints
  + And you are pretty much able to generate SQL quires to grab the exact data you are looking for.
* On the top of that I’ll show you how to integrate your SQL databases your API using two different methods.
  + Integration using raw SQL quires.
  + Integration using ORMs
  + So no matter which ultimately you prefer you have all the resources to start building out your own projects.
* We will also familiarize our self with Database Migration Tool like Alembic**.**
* Which allow us to make incremental changes to our database schema to track changes in git.
* We will also learn tools like Postman to construct http packets so we can develop our API during the whole development process.
* Author: When it comes to testing I’ve added about two to three hours of content going over how to setup Automated Integration Test.
* So that when you make changes to you code you can run these automated tests to verify that your code changes haven’t broke any pre-existing functionality.
* After testing we are going to move on the deployment phase.
* We are going to actually deploy our application including two different deployment scenarios.
* The first one is probably the most common scenario which is deploying your app onto an Ubuntu machine that can be hosted on any cloud provider like AWS, GCP, AZURE or even Digital Ocean.
* We cover things like Nginx to act as reverse proxy.
* We’ll configure our own system D service.
* We’ll setup a firewall to block all non-http traffic.
* And we’ll even setup SSL tha­­­t our application can handle HTTPS traffic.
* The second one is how you can deploy an application onto Heroku because maybe you can’t afford to pay for cloud services or maybe you don’t have the ability to sign-up for an account or something like that so I do want to make sure that you still have a way of deploying your applications so you can show off to your friends and family what you created.
* I added Heroku section because they got a very nice and convenient free tier where we can deploy our entire application for free we don’t need to sign-up with a credit card.
* So all the cool kinds are hard-core into Docker today I am going to show you to Dockerize your API in case that it is you preferred method of deployment.
* Then Finally We going to wrap things up by building out our very own CI/CD pipeline using GitHub Actions.
* This will allow us to pushout changes to GitHub resulting in our pipeline running which will pull our code run all over integration tests build all of the necessary images and if all the tests ultimately pass it will actually push our changes to production environment.
* So, that we can do all of this in an automated fashion with having to manually go in and run each step manually.

Tech Stack

* Let’s look at our Tech Stack since this is a Python API course we will be using Python to build out API.
* There were a couple of framework in python that we could have used most notably Django and Flask but I decided to use neither one of them.
* I decided to use a newer framework called FastAPI and the reason why you choose to use this framework because it has an API kind of built-in mind. It wasn’t there to address like Model View Controller type scenario.
* It really is all about building out API and on top of that it is really fast in terms of performance and makes it really easy to spin up new APIs.
* One my favourite feature of this framework is the auto documentation functionality.
* When you built an API you have to document how your API works and this is a very cumbersome task because any time you make any change in your API you have to remember to update your API with in the frontend could be making the wrong request.
* FastAPI automatically documents your API for you so you don’t have to do it yourself it’s a truly game changer.
* I choose Postgres because it is my favourite. It doesn’t really matter what type of SQL database you use they’re all fundamentally the same with only minor differences.
* For ORM we use SQLAlchemy because that’s the most popular one for Python.

Virtual Environment

* After Setting up Python and VS Code the next thing is to make virtual Environments.
* So let see what virtual environments are and what problems do they try to address.
* In short it helps by providing an isolated environment to work on projects that required a number of packages, libraries and framework as their dependencies and stay away from different version problems and other conflicts.

Installation