2025-11-26

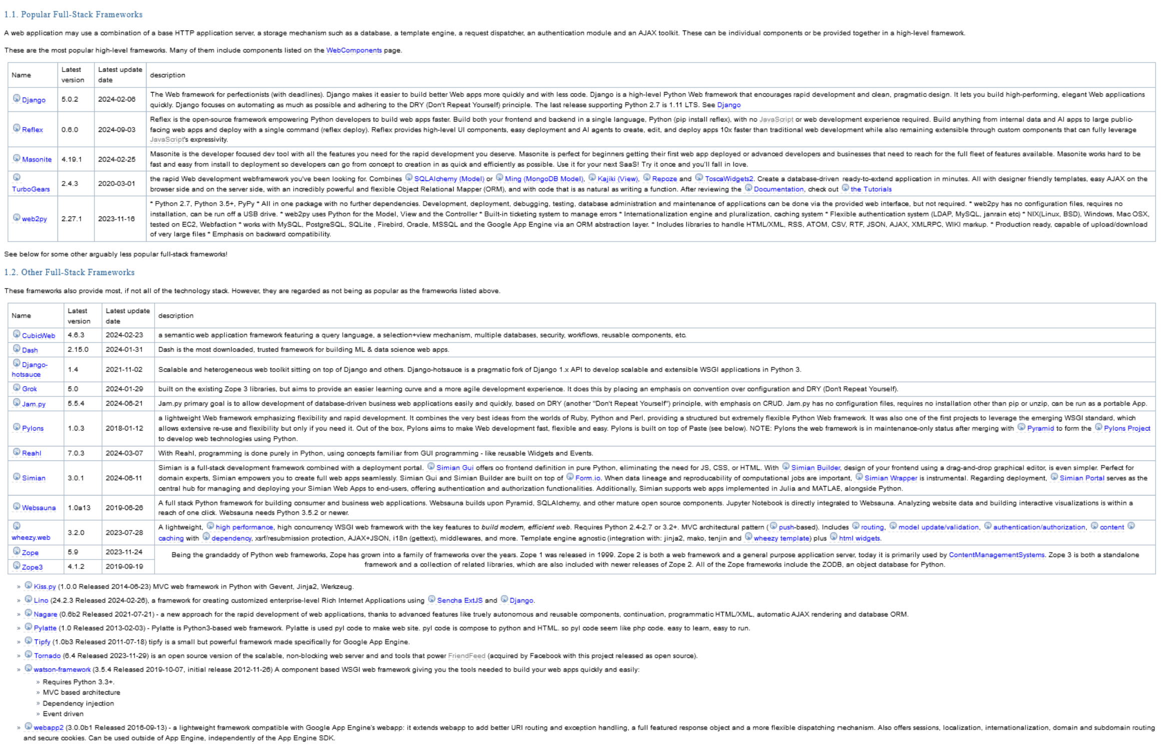
Организационные вопросы

1. **Работы по треку 2 (индивидуальные проекты)** желательно защитить **10 декабря (следующее занятие)**; иной срок **необходимо согласовать со мной заранее**
2. **Весь код по всем лабораторным нужно передать (можно репозиторий, можно 1 архив)** — прошу старост организовать централизованно сбор данных
3. **Сдавать лабораторные можно до 17 декабря включительно**:
   1. По понедельникам в корпусе с 11:00 до 13:30
   2. По понедельникам удаленно в 16:00
   3. По вторникам в корпусе с 11:00 до 15:00
   4. По средам удаленно с 18:00 (или раньше по договоренности)
   5. Кто планирует удаленно — на всякий случай прошу договориться в письме, чтобы можно было спланировать время
4. **22 декабря (понедельник) в 16:00** — подводим итоги, выставляем по результатам оценки

Веб-фреймворки в экосистеме Python

Экосистема Python уделяет значительное внимание веб-технологиям, достаточно посмотреть на официальную вики:

WebFrameworks - Python Wiki



1.1. Popular Full-Stack Frameworks

A web application may use a combination of a base HTTP application server, a storage mechanism such as a database, a template engine, a request dispatcher, an authentication module and an AJAX toolkit. These can be individual components or be provided together in a high-level framework.

These are the most popular high-level frameworks. Many of them include components listed on the WebComponents page.

Name	Latest version	Latest update date	description
Django	5.0.2	2024-02-06	The Web framework for perfectionists (with deadlines). Django makes it easier to build better Web apps more quickly and with less code. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It lets you build high-performing, elegant Web applications quickly. Django focuses on automating as much as possible and adhering to the DRY (Don't Repeat Yourself) principle. The last release supporting Python 2.7 is 1.11 LTS. See Django
Reflex	0.6.0	2024-09-03	Reflex is the open-source framework empowering Python developers to build web apps faster. Build both your frontend and backend in a single language, Python (pip install reflex), with no JavaScript or web development experience required. Build anything from internal data and AI apps to large public-facing web apps and deploy with a single command (reflex deploy). Reflex provides high-level UI components, easy deployment and AI agents to create, edit, and deploy apps 10x faster than traditional web development while also remaining extensible through custom components that can fully leverage JavaScript's expressivity.
Masonite	4.19.1	2024-02-25	Masonite is the developer focused dev tool with all the features you need for the rapid development you deserve. Masonite is perfect for beginners getting their first web app deployed or advanced developers and businesses that need to reach for the full fleet of features available. Masonite works hard to be fast and easy from install to deployment so developers can go from concept to creation in as quick and efficiently as possible. Use it for your next SaaS! Try it once and you'll fall in love.
TurboGears	2.4.3	2020-03-01	the rapid Web development webframework you've been looking for. Combines SQLAlchemy (Model) or Ming (MongoDB Model), Kajiki (View), Repoze and ToscaWidgets2. Create a database-driven ready-to-extend application in minutes. All with designer friendly templates, easy AJAX on the browser side and on the server side, with an incredibly powerful and flexible Object Relational Mapper (ORM), and with code that is as natural as writing a function. After reviewing the Documentation, check out the Tutorials
web2py	2.27.1	2023-11-16	* Python 2.7, Python 3.5+, PyPy * All in one package with no further dependencies. Development, deployment, debugging, testing, database administration and maintenance of applications can be done via the provided web interface, but not required. * web2py has no configuration files, requires no installation, can be run off a USB drive. * web2py uses Python for the Model, View and the Controller * Built-in ticketing system to manage errors * Internationalization engine and pluralization, caching system * Flexible authentication system (LDAP, MySQL, janrain etc) * NIX(Linux, BSD), Windows, Mac OSX, tested on EC2, Webfaction * works with MySQL, PostgreSQL, SQLite , Firebird, Oracle, MSSQL and the Google App Engine via an ORM abstraction layer. * Includes libraries to handle HTML/XML, RSS, ATOM, CSV, RTF, JSON, AJAX, XMLRPC, WIKI markup. * Production ready, capable of upload/download of very large files * Emphasis on backward compatibility.

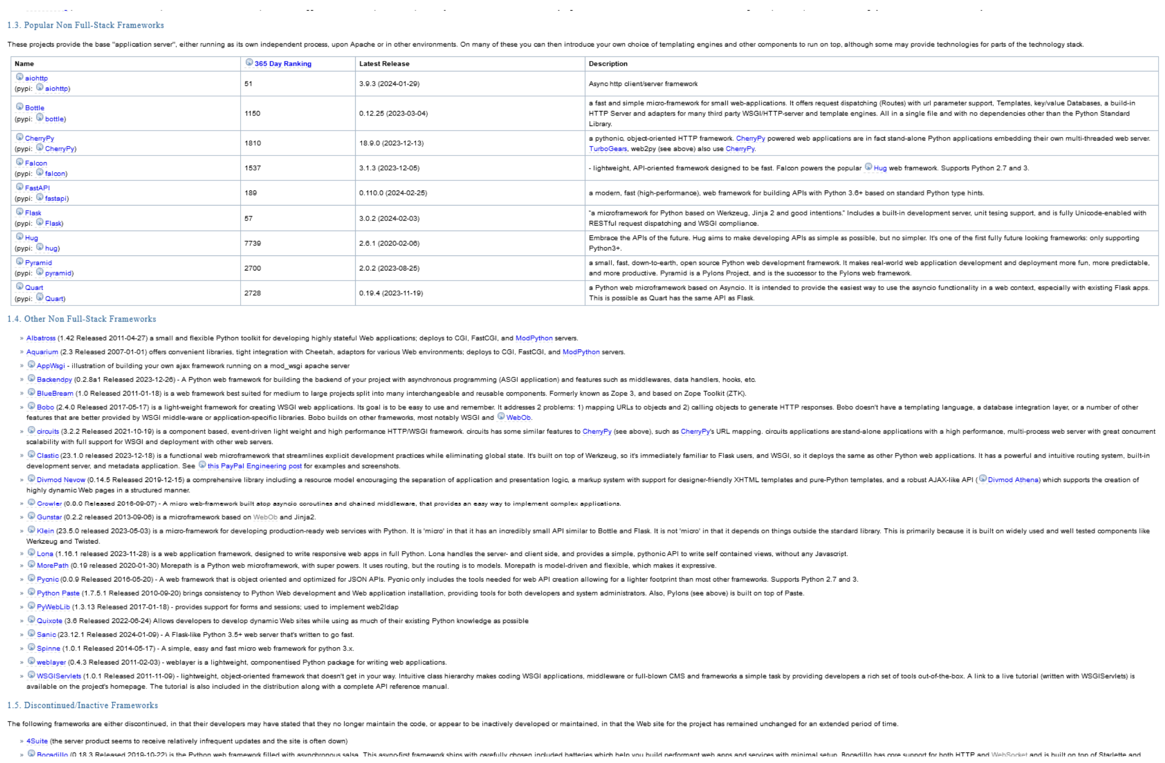
See below for some other arguably less popular full-stack frameworks!

1.2. Other Full-Stack Frameworks

These frameworks also provide most, if not all of the technology stack. However, they are regarded as not being as popular as the frameworks listed above.

Name	Latest version	Latest update date	description
CubicWeb	4.6.3	2024-02-23	a semantic web application framework featuring a query language, a selection+view mechanism, multiple databases, security, workflows, reusable components, etc.
Dash	2.15.0	2024-01-31	Dash is the most downloaded, trusted framework for building ML & data science web apps.
Django-hotsauce	1.4	2021-11-02	Scalable and heterogeneous web toolkit sitting on top of Django and others. Django-hotsauce is a pragmatic fork of Django 1.x API to develop scalable and extensible WSGI applications in Python 3.
Grok	5.0	2024-01-29	built on the existing Zope 3 libraries, but aims to provide an easier learning curve and a more agile development experience. It does this by placing an emphasis on convention over configuration and DRY (Don't Repeat Yourself).
Jam.py	5.5.4	2024-06-21	Jam.py primary goal is to allow development of database-driven business web applications easily and quickly, based on DRY (another "Don't Repeat Yourself") principle, with emphasis on CRUD. Jam.py has no configuration files, requires no installation other than pip or unzip, can be run as a portable App.
Pylons	1.0.3	2018-01-12	a lightweight Web framework emphasizing flexibility and rapid development. It combines the very best ideas from the worlds of Ruby, Python and Perl, providing a structured but extremely flexible Python Web framework. It was also one of the first projects to leverage the emerging WSGI standard, which allows extensive re-use and flexibility but only if you need it. Out of the box, Pylons aims to make Web development fast, flexible and easy. Pylons is built on top of Paste (see below). NOTE: Pylons the web framework is in maintenance-only status after merging with Pyramid to form the Pylons Project to develop web technologies using Python.
Reahl	7.0.3	2024-03-07	With Reahl, programming is done purely in Python, using concepts familiar from GUI programming - like reusable Widgets and Events.
Simian	3.0.1	2024-06-11	Simian is a full-stack development framework combined with a deployment portal. Simian Gui offers oo frontend definition in pure Python, eliminating the need for JS, CSS, or HTML. With Simian Builder, design of your frontend using a drag-and-drop graphical editor, is even simpler. Perfect for domain experts, Simian empowers you to create full web apps seamlessly. Simian Gui and Simian Builder are built on top of Form.io. When data lineage and reproducibility of computational jobs are important, Simian Wrapper is instrumental. Regarding deployment, Simian Portal serves as the central hub for managing and deploying your Simian Web Apps to end-users, offering authentication and authorization functionalities. Additionally, Simian supports web apps implemented in Julia and MATLAE, alongside Python.
Websauna	1.0a13	2019-06-26	A full stack Python framework for building consumer and business web applications. Websauna builds upon Pyramid, SQLAlchemy, and other mature open source components. Jupyter Notebook is directly integrated to Websauna. Analyzing website data and building interactive visualizations is within a reach of one click. Websauna needs Python 3.5.2 or newer.
wheezy.web	3.2.0	2023-07-28	A lightweight, high performance, high concurrency WSGI web framework with the key features to *build modern, efficient web*. Requires Python 2.4-2.7 or 3.2+. MVC architectural pattern (push-based). Includes routing, model update/validation, authentication/authorization, content caching with dependency, xsrf/resubmission protection, AJAX+JSON, i18n (gettext), middlewares, and more. Template engine agnostic (integration with: jinja2, mako, tenjin and wheezy template) plus html widgets.
Zope	5.9	2023-11-24	Being the grandaddy of Python web frameworks, Zope has grown into a family of frameworks over the years. Zope 1 was released in 1999. Zope 2 is both a web framework and a general purpose application server, today it is primarily used by ContentManagementSystems. Zope 3 is both a standalone framework and a collection of related libraries, which are also included with newer releases of Zope 2. All of the Zope frameworks include the ZODB, an object database for Python.
Zope3	4.1.2	2019-09-19	

- Kiss.py (1.0.0 Released 2014-06-23) MVC web framework in Python with Gevent, Jinja2, Werkzeug.
- Lino (24.2.3 Released 2024-02-26), a framework for creating customized enterprise-level Rich Internet Applications using Sencha ExtJS and Django.
- Nagare (0.6b2 Released 2021-07-21) - a new approach for the rapid development of web applications, thanks to advanced features like truely autonomous and reusable components, continuation, programmatic HTML/XML, automatic AJAX rendering and database ORM.
- Pylatte (1.0 Released 2013-02-03) - Pylatte is Python3-based web framework. Pylatte is used pyl code to make web site. pyl code is compose to python and HTML. so pyl code seem like php code. easy to learn, easy to run.
- Tipfy (1.0b3 Released 2011-07-18) tipfy is a small but powerful framework made specifically for Google App Engine.
- Tornado (6.4 Released 2023-11-29) is an open source version of the scalable, non-blocking web server and and tools that power FriendFeed (acquired by Facebook with this project released as open source).
- watson-framework (3.5.4 Released 2019-10-07, initial release 2012-11-26) A component based WSGI web framework giving you the tools needed to build your web apps quickly and easily:
  - Requires Python 3.3+.
  - MVC based architecture
  - Dependency injection
  - Event driven
- webapp2 (3.0.0b1 Released 2016-09-13) - a lightweight framework compatible with Google App Engine's webapp: it extends webapp to add better URI routing and exception handling, a full featured response object and a more flexible dispatching mechanism. Also offers sessions, localization, internationalization, domain and subdomain routing and secure cookies. Can be used outside of App Engine, independently of the App Engine SDK.



1.3. Popular Non Full-Stack Frameworks

These projects provide the base "application server", either running as its own independent process, upon Apache or in other environments. On many of these you can then introduce your own choice of templating engines and other components to run on top, although some may provide technologies for parts of the technology stack.

Name	365 Day Ranking	Latest Release	Description
aiohttp (pypi: aiohttp)	51	3.9.3 (2024-01-29)	Async http client/server framework
Bottle (pypi: bottle)	1150	0.12.25 (2023-03-04)	a fast and simple micro-framework for small web-applications. It offers request dispatching (Routes) with url parameter support, Templates, key/value Databases, a build-in HTTP Server and adapters for many third party WSGI/HTTP-server and template engines. All in a single file and with no dependencies other than the Python Standard Library.
CherryPy (pypi: CherryPy)	1810	18.9.0 (2023-12-13)	a pythonic, object-oriented HTTP framework. CherryPy powered web applications are in fact stand-alone Python applications embedding their own multi-threaded web server. TurboGears, web2py (see above) also use CherryPy.
Falcon (pypi: falcon)	1537	3.1.3 (2023-12-05)	- lightweight, API-oriented framework designed to be fast. Falcon powers the popular Hug web framework. Supports Python 2.7 and 3.
FastAPI (pypi: fastapi)	189	0.110.0 (2024-02-25)	a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints.
Flask (pypi: Flask)	57	3.0.2 (2024-02-03)	"a microframework for Python based on Werkzeug, Jinja 2 and good intentions." Includes a built-in development server, unit tesing support, and is fully Unicode-enabled with RESTful request dispatching and WSGI compliance.
Hug (pypi: hug)	7739	2.6.1 (2020-02-06)	Embrace the APIs of the future. Hug aims to make developing APIs as simple as possible, but no simpler. It's one of the first fully future looking frameworks: only supporting Python3+.
Pyramid (pypi: pyramid)	2700	2.0.2 (2023-08-25)	a small, fast, down-to-earth, open source Python web development framework. It makes real-world web application development and deployment more fun, more predictable, and more productive. Pyramid is a Pylons Project, and is the successor to the Pylons web framework.
Quart (pypi: Quart)	2728	0.19.4 (2023-11-19)	a Python web microframework based on Asyncio. It is intended to provide the easiest way to use the asyncio functionality in a web context, especially with existing Flask apps. This is possible as Quart has the same API as Flask.

1.4. Other Non Full-Stack Frameworks

» Albatross (1.42 Released 2011-04-27) a small and flexible Python toolkit for developing highly stateful Web applications; deploys to CGI, FastCGI, and ModPython servers.
» Aquarium (2.3 Released 2007-01-01) offers convenient libraries, tight integration with Cheetah, adaptors for various Web environments; deploys to CGI, FastCGI, and ModPython servers.
» AppWsgi - illustration of building your own ajax framework running on a mod_wsgi apache server
» Backendpy (0.2.8a1 Released 2023-12-26) - A Python web framework for building the backend of your project with asynchronous programming (ASGI application) and features such as middlewares, data handlers, hooks, etc.
» BlueBream (1.0 Released 2011-01-18) is a web framework best suited for medium to large projects split into many interchangeable and reusable components. Formerly known as Zope 3, and based on Zope Toolkit (ZTK).
» Bobo (2.4.0 Released 2017-05-17) is a light-weight framework for creating WSGI web applications. Its goal is to be easy to use and remember. It addresses 2 problems: 1) mapping URLs to objects and 2) calling objects to generate HTTP responses. Bobo doesn't have a templating language, a database integration layer, or a number of other features that are better provided by WSGI middle-ware or application-specific libraries. Bobo builds on other frameworks, most notably WSGI and WebOb.
» circuits (3.2.2 Released 2021-10-19) is a component based, event-driven light weight and high performance HTTP/WSGI framework. circuits has some similar features to CherryPy (see above), such as CherryPy's URL mapping. circuits applications are stand-alone applications with a high performance, multi-process web server with great concurrent scalability with full support for WSGI and deployment with other web servers.
» Clastic (23.1.0 released 2023-12-18) is a functional web microframework that streamlines explicit development practices while eliminating global state. It's built on top of Werkzeug, so it's immediately familiar to Flask users, and WSGI, so it deploys the same as other Python web applications. It has a powerful and intuitive routing system, built-in development server, and metadata application. See this PayPal Engineering post for examples and screenshots.
» Divmod Nevow (0.14.5 Released 2019-12-15) a comprehensive library including a resource model encouraging the separation of application and presentation logic, a markup system with support for designer-friendly XHTML templates and pure-Python templates, and a robust AJAX-like API (Divmod Athena) which supports the creation of highly dynamic Web pages in a structured manner.
» Crowler (0.0.0 Released 2016-09-07) - A micro web-framework built atop asyncio coroutines and chained middleware, that provides an easy way to implement complex applications.
» Gunstar (0.2.2 released 2013-09-06) is a microframework based on WebOb and Jinja2.
» Klein (23.5.0 released 2023-05-03) is a micro-framework for developing production-ready web services with Python. It is 'micro' in that it has an incredibly small API similar to Bottle and Flask. It is not 'micro' in that it depends on things outside the standard library. This is primarily because it is built on widely used and well tested components like Werkzeug and Twisted.
» Lona (1.16.1 released 2023-11-28) is a web application framework, designed to write responsive web apps in full Python. Lona handles the server- and client side, and provides a simple, pythonic API to write self contained views, without any Javascript.
» MorePath (0.19 released 2020-01-30) Morepath is a Python web microframework, with super powers. It uses routing, but the routing is to models. Morepath is model-driven and flexible, which makes it expressive.
» Pycnic (0.0.9 Released 2016-05-20) - A web framework that is object oriented and optimized for JSON APIs. Pycnic only includes the tools needed for web API creation allowing for a lighter footprint than most other frameworks. Supports Python 2.7 and 3.
» Python Paste (1.7.5.1 Released 2010-09-20) brings consistency to Python Web development and Web application installation, providing tools for both developers and system administrators. Also, Pylons (see above) is built on top of Paste.
» PyWebLib (1.3.13 Released 2017-01-18) - provides support for forms and sessions; used to implement web2ldap
» Quixote (3.6 Released 2022-06-24) Allows developers to develop dynamic Web sites while using as much of their existing Python knowledge as possible
» Sanic (23.12.1 Released 2024-01-09) - A Flask-like Python 3.5+ web server that's written to go fast.
» Spinne (1.0.1 Released 2014-05-17) - A simple, easy and fast micro web framework for python 3.x.
» weblayer (0.4.3 Released 2011-02-03) - weblayer is a lightweight, componentised Python package for writing web applications.
» WSGIServlets (1.0.1 Released 2011-11-09) - lightweight, object-oriented framework that doesn't get in your way. Intuitive class hierarchy makes coding WSGI applications, middleware or full-blown CMS and frameworks a simple task by providing developers a rich set of tools out-of-the-box. A link to a live tutorial (written with WSGIServlets) is available on the project's homepage. The tutorial is also included in the distribution along with a complete API reference manual.

1.5. Discontinued/Inactive Frameworks

The following frameworks are either discontinued, in that their developers may have stated that they no longer maintain the code, or appear to be inactively developed or maintained, in that the Web site for the project has remained unchanged for an extended period of time.

» 4Suite (the server product seems to receive relatively infrequent updates and the site is often down)
» Bocadillo (0.18.3 Released 2019-10-22) is the Python web framework filled with asynchronous salsa. This async-first framework ships with carefully chosen included batteries which help you build performant web apps and services with minimal setup. Bocadillo has core support for both HTTP and WebSocket and is built on top of Starlette and

Django

Сайт: The web framework for perfectionists with deadlines | Django

Документация: Django documentation | Django documentation | Django

Исходники: GitHub - django/django: The Web framework for perfectionists with deadlines.

```
pip install Django
```

Django · PyPI

Актуальная версия: Django 5.2.8 (LTS), Django 6.0 ожидается в декабре 2025. Требует Python ≥3.10.

- Работает по принципу «все включено», особенно в части структур данных
- Включает собственную модель ORM (Object-Relational Mapping) для управления базами данных, аутентификации, роутинга, шаблонов (Django ORM Tutorial - The concept to master Django framework - DataFlair)

```
from django.db import models

class Author(models.Model):
    name = models.CharField(max_length=255)
    email = models.EmailField(unique=True)
    created_on = models.DateTimeField(auto_now_add=True)
    last_logged_in = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.name
```

```
from django.contrib.auth.models import Author

a = Author(name="John Doe", email="johndoe@example.com")
a.save()

authors = Author.objects.filter(active=True).order_by("-created_on")[:5]
for author in authors:
    print(author.name)
# John Doe (created 2022-01-01)
# Jane Smith (created 2022-01-05)
```

- делает акцент на безопасности
- из коробки поддерживает масштабирование и в целом направлен на обеспечение гибкости при балансировки нагрузки
- философски старается придерживаться принципа DRY (Don't Repeat Yourself)
- **Новое в 6.0 (доступно частично с 5.2)**: поддержка template partials (шаблонных фрагментов), улучшенная интеграция с HTMX/Alpine.js как альтернатива SPA-архитектуре
- НО! для небольших проектов может быть избыточен и с более высоким порогом вхождения

Django часто являлся представителем Питона в одном из вариантов LAMP-стека (Linux, Apache, MySQL, Python/PHP/Perl)

Flask

Сайт: [Welcome to Flask — Flask Documentation (3.1.x)]()

Документация: Tutorial — Flask Documentation (3.1.x) / Quickstart — Flask Documentation (3.1.x)

Исходники: GitHub - pallets/flask: The Python micro framework for building web applications.

Flask · PyPI

```
pip install Flask
```

- один из первых популярных микрофреймворков для Питона (минимум обвеса из коробки, легко достраивается до своих нужд, легко дополняется модулями/плагинами и в целом мало весит)
- поддерживает шаблонизацию Jinja2 (Jinja — Jinja Documentation (3.1.x))
- простота начала работы над проектом (легко получить работающее приложение) и низкая кривая обучения
- модульность подталкивает к микросервисной архитектуре
- **Новое в 3.1.2+:** поддержка асинхронных view-функций и обработчиков (async views), поддержка Python 3.13
- НО! Многие нужные вещи (SQL, CORS, OAuth2) доставляются в виде отдельных модулей, которые разрабатывают часто не те же люди

```
from flask import Flask

app = Flask(__name__)

from markupsafe import escape

@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return f'User {escape(username)}'

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return f'Post {post_id}'

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    # show the subpath after /path/
    return f'Subpath {escape(subpath)}'

@app.get('/overview')
def show_overview():
    return flask.render_template('overview.html',
user=users[get_username()]['name'])

@app.post('/upload')
def upload_xls():
    polyex_data.upload(flask.request.get_data())
    return flask.jsonify(polyex_data.get_dashboard_data())

@app.get('/chart/<code>')
def get_chart_data(code:str):
    return flask.jsonify(polyex_data.get_price_graph(code))
```

FastAPI

Сайт: [FastAPI]()

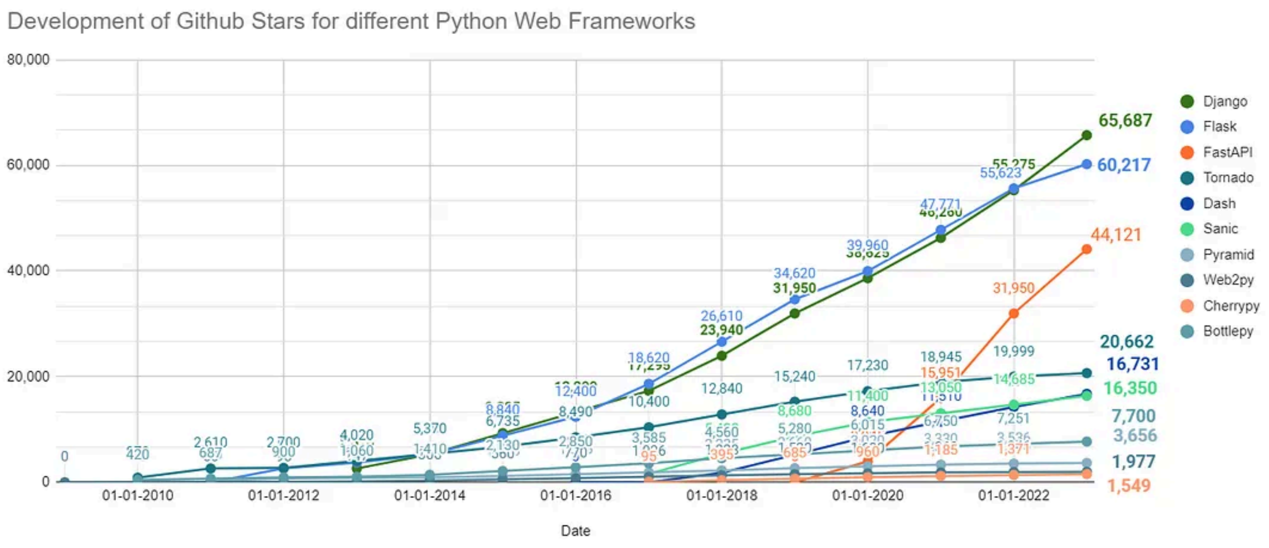
Документация: [Tutorial - User Guide - FastAPI]()

Исходники: [GitHub - fastapi/fastapi: FastAPI framework, high performance, easy to learn, fast to code, ready for production]()

[fastapi · PyPI]()

```
pip install fastapi[standard]
```

Поддерживает Python 3.10+ (рекомендуется 3.12), включая Python 3.14.



- полностью асинхронный фреймворк на уровне концепции
- использует `pydantic` [Welcome to Pydantic - Pydantic](https://docs.pydantic.dev) для движка моделей данных
- **один из самых популярных Python-фреймворков в 2025 году** — идеален для микросервисов, AI/ML workflows, backend SaaS-решений

```
import uvicorn
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def home():
    return {"Hello": "World"}

if __name__ == "__main__":
    uvicorn.run("fastapi_code:app")
```

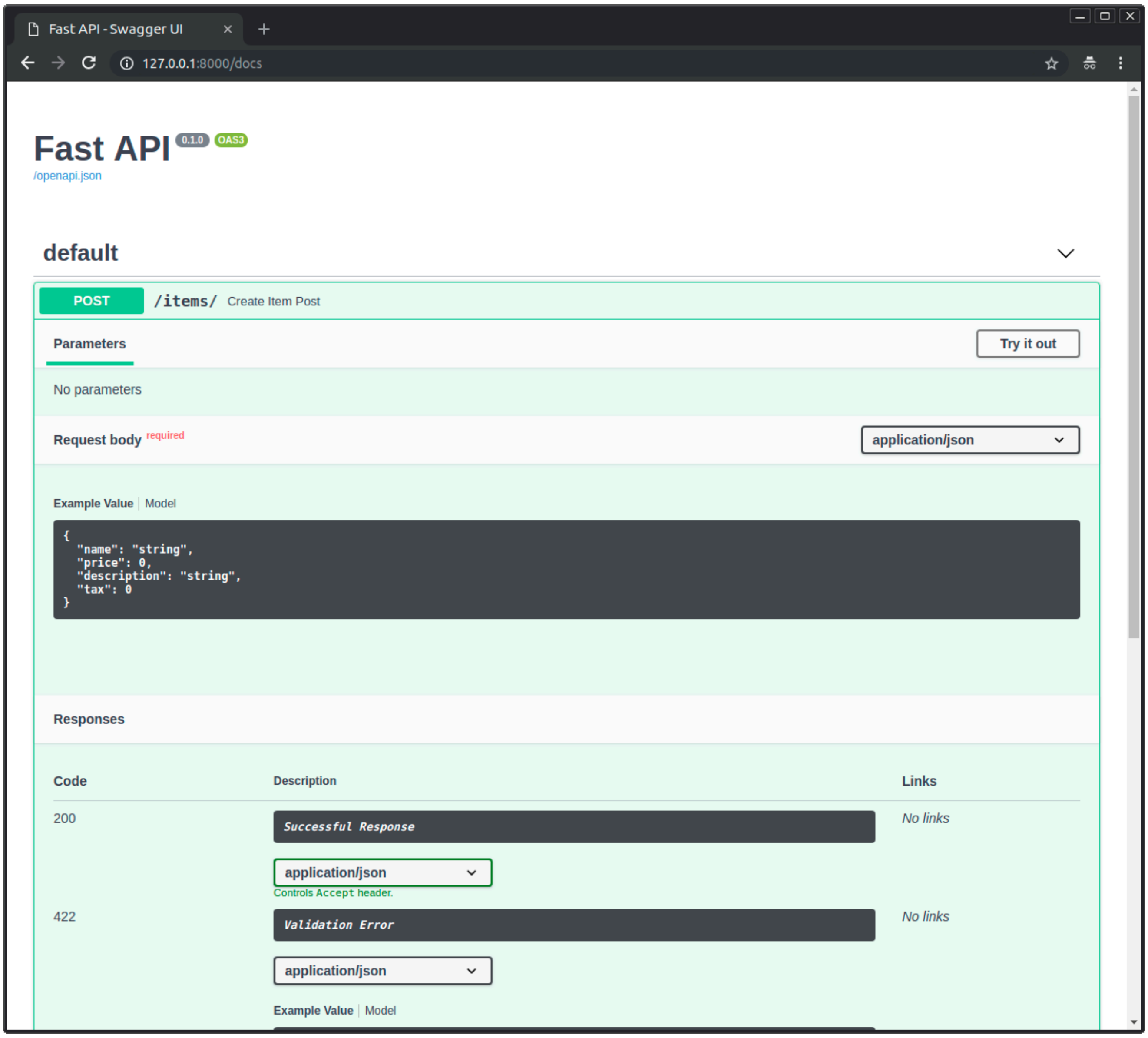
```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Request(BaseModel):
    username: str
    password: str

@app.post("/login")
async def login(req: Request):
    if req.username == "testdriven.io" and req.password == "testdriven.io":
        return {"message": "success"}
    return {"message": "Authentication Failed"}

# correct payload format
```

```
╳ curl -X POST 'localhost:8000/login' \
    --header 'Content-Type: application/json' \
    --data-raw '{\"username\":
\"testdriven.io\",\"password\":\"testdriven.io\"}'

{"message":"success"}

# incorrect payload format
╳ curl -X POST 'localhost:8000/login' \
    --header 'Content-Type: application/json' \
    --data-raw '{\"username\":
\"testdriven.io\",\"passwords\":\"testdriven.io\"}'

{"detail":[{"loc":["body","password"],"msg":"field
required","type":"value_error.missing"}]}

from pydantic import BaseModel

app = FastAPI()

class Request(BaseModel):
    username: str
    email: str
    password: str

class Response(BaseModel):
    username: str
    email: str

@app.post("/login", response_model=Response)
async def login(req: Request):
    if req.username == "testdriven.io" and req.password == "testdriven.io":
        return req
    return {"message": "Authentication Failed"}

# output
╳ curl -X POST 'localhost:8000/login' \
    --header 'Content-Type: application/json' \
    --data-raw
'{\"username\":\"testdriven.io\",\"email\":\"admin@testdriven.io\",\"passwo
rd\":\"testdriven.io\"}'

{"username":"testdriven.io","email":"admin@testdriven.io"}
```

```
from fastapi import BackgroundTasks

def process_file(filename: str):
    # process file :: takes minimum 3 secs (just an example)
    pass

def write_notification(email: str, message=""):
    with open("log.txt", mode="w") as email_file:
        content = f"notification for {email}: {message}"
        email_file.write(content)

@app.post("/upload/{filename}")
async def upload_and_process(filename: str, background_tasks:
BackgroundTasks):
    background_tasks.add_task(process_file, filename)
    return {"message": "processing file"}

# мониторить их можно через встроенный эндпойнт /metrics
```

> **Note**
>
> Во Flask нам бы понадобилось использовать для такого Celery + Redis (Asynchronous Tasks with Flask and Celery | TestDriven.io)

- генерирует встроенную Swagger/ReDoc-документацию


Fast API - Swagger UI

127.0.0.1:8000/docs

Fast API 0.1.0 OAS3

/openapi.json

default

POST /items/ Create Item Post

Parameters

No parameters

Request body required

application/json

Example Value | Model

```
{
  "name": "string",
  "price": 0,
  "description": "string",
  "tax": 0
}
```

Responses

Code	Description	Links
200	Successful Response application/json Controls Accept header.	No links
422	Validation Error application/json Example Value | Model	No links

- для SQL использует SQLModel, которая построена поверх SQLAlchemy + Pydantic

```python
from typing import Optional

from sqlmodel import Field, SQLModel


class Hero(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    name: str
    secret_name: str
    age: Optional[int] = None

hero_1 = Hero(name="Deadpond", secret_name="Dive Wilson")
hero_2 = Hero(name="Spider-Boy", secret_name="Pedro Parqueador")
hero_3 = Hero(name="Rusty-Man", secret_name="Tommy Sharp", age=48)
```

- НО! Требует разработки фронтенда!

Litestar

Сайт: [Litestar | Effortlessly Build Performant APIs]

Документация: [Litestar library documentation — Litestar Framework]

Исходники: [GitHub - litestar-org/litestar]

```
pip install litestar
```

- высокопроизводительный асинхронный ASGI-фреймворк с акцентом на типобезопасность
- альтернатива FastAPI с **меньшим потреблением памяти** (использует `msgspec` ([GitHub - jcrist/msgspec: A fast serialization and validation library, with builtin support for JSON, MessagePack, YAML, and TOML]) вместо Pydantic)
- мощная система dependency injection на всех уровнях приложения (как в FastAPI)
- автоматическая генерация OpenAPI-документации (Swagger, Redoc, Stoplight Elements) — как в FastAPI
- встроенная поддержка WebSockets, JWT-аутентификации, кэширования, rate-limiting

```python
from litestar import Litestar, get

@get("/greet")
async def greet(name: str) -> str:
    return f"Hi, {name}!"

app = Litestar([greet])
```

- НО! Меньше сообщество и экосистема плагинов по сравнению с FastAPI

Tornado

Сайт + документация: [Tornado Web Server — Tornado 6.5.2 documentation]()

Исходники: [GitHub - tornadoweb/tornado: Tornado is a Python web framework and asynchronous networking library, originally developed at FriendFeed.]()

[tornado · PyPI]()

```
pip install tornado
```

Актуальная версия: Tornado 6.5.2. Требует Python ≥3.9, поддерживает Python 3.14 и экспериментально free-threading режим Python 3.13.

- один из первых фреймворков, поставивших асинхронность в приоритет
- поддержка веб-сокетов
- встроенная поддержка для HTTP/1.1 (по тем временам)

```python
import asyncio
import tornado

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")

def make_app():
    return tornado.web.Application([(r"/", MainHandler)])

async def main():
    app = make_app()
    app.listen(8888)
    await asyncio.Event().wait()

if __name__ == "__main__":
    asyncio.run(main())
```

Фреймворки для визуализации данных, дашбордов и ML/AI

Streamlit

Сайт: Streamlit • A faster way to build and share data apps

Документация: Streamlit documentation

Исходники: GitHub - streamlit/streamlit

streamlit · PyPI

```
pip install streamlit
```

- позволяет быстро превращать Python-скрипты в интерактивные веб-приложения
- идеален для быстрого прототипирования, создания дашбордов, демонстрации ML-моделей
- декларативный синтаксис
- встроенные компоненты для визуализации данных (интеграция с Matplotlib, Plotly, Altair и др.)
- **Streamlit Community Cloud** — бесплатное развертывание приложений

```
import streamlit as st
import pandas as pd
import numpy as np

st.title('Мой первый Streamlit-дашборд')

chart_data = pd.DataFrame(
    np.random.randn(20, 3),
    columns=['a', 'b', 'c'])

st.line_chart(chart_data)

option = st.selectbox('Выберите значение:', ['A', 'B', 'C'])
st.write('Вы выбрали:', option)
```

НО! При каждом взаимодействии пользователя скрипт перезапускается полностью, может замедлять сложные приложения. Ограниченные возможности кастомизации UI.

Gradio

gradio

Сайт: Gradio

Документация: Gradio Docs

Исходники: GitHub - gradio-app/gradio

gradio · PyPI

```
pip install gradio
```

Актуальная версия (ноябрь 2025): Gradio 5.x — **полностью переработанная версия с SSR (Server-Side Rendering)**.

- библиотека для создания веб-интерфейсов к ML-моделям за несколько строк кода
- бесшовная интеграция с TensorFlow, PyTorch, Hugging Face
- развертывание на Hugging Face Spaces
- встроенная поддержка streaming для аудио/видео (FastRTC)
- улучшенный дизайн, enterprise-уровень безопасности (аудит Trail of Bits), SSR для мгновенной загрузки (с 5 версии)

```python
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * int(intensity)

demo = gr.Interface(
    fn=greet,
    inputs=["text", gr.Slider(value=2, minimum=1, maximum=10, step=1)],
    outputs=["text"],
)

demo.launch()
```

НО! Оптимизирован для демо ML-моделей; для сложных приложений лучше использовать другие фреймворки.

Dash (Plotly)

Сайт: Dash Documentation & User Guide | Plotly

Документация: Dash in 20 Minutes Tutorial

Исходники: GitHub - plotly/dash

dash · PyPI

```
pip install dash
```

- самый популярный фреймворк для создания дашбордов
- построен поверх Plotly.js, React и Flask
- поддержка ~50 типов интерактивных графиков, включая карты
- callback-система для реактивного обновления компонентов
- **Dash Enterprise** для корпоративного развертывания с OAuth, мониторингом, масштабированием

```python
from dash import Dash, dcc, html, Input, Output

app = Dash(__name__)

app.layout = html.Div([
    html.H1("Hello Dash"),
    dcc.Input(id='input-box', value='initial value', type='text'),
    html.Div(id='output-div')
])

@app.callback(
    Output('output-div', 'children'),
    Input('input-box', 'value')
)
def update_output(value):
    return f'Value: {value}'

if __name__ == '__main__':
    app.run(debug=True)
```

НО! Требует понимания callback-архитектуры; HTML-подобный синтаксис компонентов.

Bokeh

Сайт: Bokeh

Документация: Bokeh Documentation

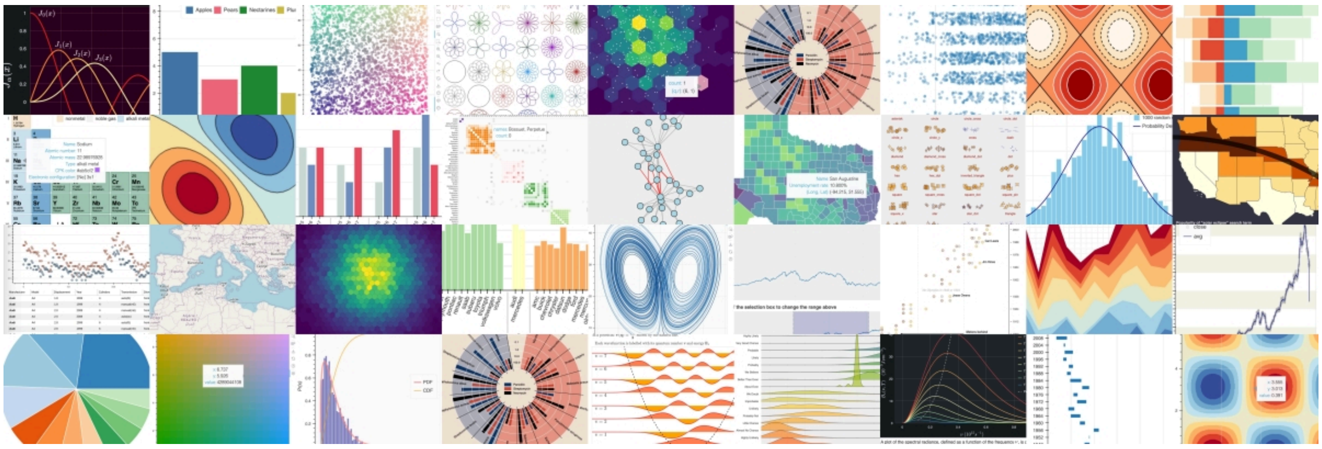
Исходники: GitHub - bokeh/bokeh

bokeh · PyPI

```
pip install bokeh
```

- библиотека для создания интерактивных визуализаций в браузере
- рендеринг через HTML/JavaScript — плавная работа даже с большими датасетами
- встроенная поддержка streaming data (идеально для мониторинга, реал-тайм дашбордов)
- интеграция с pandas, NumPy, SciPy
- возможность встраивания в веб-приложения на Flask, Django, FastAPI

```
from bokeh.plotting import figure, output_file, show

# instantiating the figure object
graph = figure(title="Bokeh Line Graph")

# the points to be plotted
x = [1, 2, 3, 4, 5]
y = [5, 4, 3, 2, 1]

# plotting the line graph
graph.line(x, y)

# displaying the model
show(graph)
```



НО! Для создания полноценных приложений требуется дополнительный веб-фреймворк.

Panel (HoloViz)

Сайт: [Panel — Overview]

Документация: [Panel Documentation]

Исходники: [GitHub - holoviz/panel]

[panel · PyPI]

```
pip install panel
```

- универсальный фреймворк для создания дашбордов и приложений полностью на Python
- интеграция с множеством библиотек визуализации: Bokeh, Plotly, Matplotlib, HoloViews, Altair, PyVista и т.п.
- поддержка ipywidgets
- вывод в веб-приложение (Tornado, Flask, Django, FastAPI), standalone client-side (Pyodide/PyScript), Jupyter Notebook, статичные HTML/PNG/GIF

```
import panel as pn

pn.extension()

slider = pn.widgets.IntSlider(value=5, start=1, end=5)
```

```
def model(n=5):
    return "⭐" * n

interactive_model = pn.bind(model, n=slider)

layout = pn.Column(slider, interactive_model)
layout.servable()
```

НО! Более высокий порог вхождения по сравнению со Streamlit.

Taipy

Сайт: [Taipy — Build Python Data & BI web applications]()

Документация: [Taipy Documentation]()

Исходники: [GitHub - Avaiga/taipy]()

```
pip install taipy
```

- фреймворк для корпоративных production-ready data/AI приложений
- визуальный редактор пайплайнов Taipy Studio (расширение VS Code)
- встроенное управление ML-пайплайнами
- оптимизирован для больших данных (кэширование)
- интеграция с Databricks, Dataiku, AWS SageMaker, IBM Watson и другими коммерческими ресурсами для больших данных

```
from taipy.gui import Gui

name = "World"

page = """
# Hello *<|{name}|>*!
```

```
<|{name}|input|>
"""

Gui(page).run()
```

НО! Менее известен, чем Streamlit/Dash; меньше готовых примеров в сообществе.

Фуллстэк

Reflex

Сайт: [Reflex · Web apps in Pure Python]

Документация: [Reflex Documentation]

Исходники: [GitHub - reflex-dev/reflex]

```
pip install reflex
```

- создание полноценных веб-приложений на чистом Python
- автоматическая компиляция в React/Next.js frontend
- реактивное управление состоянием через WebSockets
- интеграция с React и Vue.js компонентами
- НО! Относительно новый фреймворк; меньше production-примеров.

```
import reflex as rx

class State(rx.State):
    count: int = 0

    def increment(self):
        self.count += 1

def index():
    return rx.vstack(
        rx.heading(State.count),
        rx.button("Increment", on_click=State.increment),
    )

app = rx.App()
app.add_page(index)
```

NiceGUI

Сайт: [NiceGUI]

Документация: NiceGUI Documentation

Исходники: GitHub - zauberzeug/nicegui

```
pip install nicegui
```

- фреймворк для создания веб-интерфейсов на Python (без JS/HTML/CSS)
- построен на FastAPI (backend) + Vue.js/Quasar (frontend)
- работает как веб-приложение или в native-режиме
- поддержка 3D-сцен, графиков, работа с изображениями/видео

```
from nicegui import ui

ui.label('Hello NiceGUI!')
ui.button('Click me', on_click=lambda: ui.notify('Button clicked!'))

ui.run()
```

НО! Для сложных приложений может потребоваться кастомизация.

Также фреймворки для ознакомления

- Bottle (Bottle: Python Web Framework — Bottle 0.14-dev documentation)
- CherryPy (CherryPy — A Minimalist Python Web Framework — CherryPy 18.10.1.dev53+g8fbdef0 documentation)
- Pyramid (Welcome to Pyramid, a Python Web Framework)
- Grok grok · PyPI, GitHub - zopefoundation/grok: Grok: Now even cavemen can use Zope 3!
- Sanic (Sanic User Guide - The lightning-fast asynchronous Python web framework) — версия 25.3.0, Python 3.9+, встроенный REPL, поддержка websockets v14
- Falcon (Falcon | The minimal, fast, and secure web framework for Python)

Лабораторная работа №6

Взять ЛР №5 и любой из понравившихся веб-фреймворков и сделать веб-приложение для RAG на базе ЛР №5