

راهنمای محیط توسعه

سیستم مدیریت پروژه خودکار

راهنمای جامع برای راه اندازی و نگهداری محیط توسعه، شامل نیازمندی های سیستم، ابزارها، پیکربندی، تست و بهترین روش ها



تست



ابزارها



توسعه




راه اندازی

نیازمندی‌های سیستم


سیستم‌عامل‌های پشتیبانی

شده



Linux: اوبونتو 20.04، CentOS 8، دبیان 10


macOS: 10.15 (کاتالینا یا جدیدتر) 

Windows: ویندوز 10 (نسخه 19041+) با WSL2 

</> نیازمندی‌های نرم‌افزاری

نرم‌افزار	نسخه	کاربرد
Python	+3.8	محیط اجرایی اصلی
Node.js	+x.14	ابزارهای frontend
Git	+2.20	کنترل نسخه
Docker	+20.10	کانتینرسازی
VS Code	+1.60	محیط توسعه

🔧 نیازمندی‌های سخت‌افزاری

قطعه	حداقل	توصیه شده
پردازنده	دو هسته‌ای 2.0 گیگاهرتز	چهار هسته‌ای 3.0 گیگاهرتز
حافظه RAM	4 گیگابایت	8 گیگابایت
حافظه ذخیره‌سازی	10 گیگابایت فضای آزاد	50 گیگابایت SSD
شبکه	اتصال پهن باند	اینترنت پرسرعت

راه اندازی محیط

فرآیند راه اندازی پروژه

1 کلون کردن مخزن

دریافت کد منبع پروژه

```
git clone https://github.com/projectmanagement/autoprojectmanagement.git
```

2 ایجاد محیط مجازی

جداسازی وابستگی‌های پروژه

```
python3 -m venv venv && source venv/bin/activate
```

3 نصب وابستگی‌ها

نصب بسته‌های مورد نیاز

```
pip install -r requirements.txt -r requirements-dev.txt
```

4 راه اندازی خودکار محیط

اجرای راه اندازی خودکار

```
python management.setup_auto_environment --verbose
```

نصب پیش‌نیازها

1 محیط Python

نصب Python 3.8+ و pip

```
python3 --version
```

2 نصب Node.js

استفاده از Node Version Manager (توصیه شده)

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash
```

3 تأیید نصب

بررسی نسخه ابزارهای نصب شده

```
node --version && npm --version
```

ابزارهای توسعه

</> پیکربندی IDE

Python



پشتیبانی از زبان و دیباگ

```
ext install ms-python.python
```

Pylance



بررسی نوع و IntelliSense

```
ext install ms-python.vscode-pylance
```

Black Formatter



قالب‌بندی و سبک کد

```
ext install ms-python.black-formatter
```

GitLens



یکپارچه‌سازی Git

```
ext install eamodio.gitlens
```

Docker



پشتیبانی از کانینر

```
ext install ms-azuretools.vscode-docker
```

تنظیمات VS Code

```
{python.defaultInterpreterPath: "./venv/bin/python", "python.formatting.provider": "black", "editor.formatOnSave": true}
```

راهنمای

محیط توسعه

محیط توسعه کد

```
python:3.9-slim
WORKDIR /app
nodejs npm
dependencies*.txt
scripts-dev.txt
```

Compose

راه‌اندازی چندگانه

```
version: '3.8'
services:
  app:
    build: .
    ports:
      - 8000
    volumes:
      - ./app:/app
  redis:
    image: alpine
  postgres:
    image: postgres:13
```

شروع



اجرای

1 up

ساختار پروژه

🏗️ ساختار ماژول

services 🏠

یکپارچه‌سازی‌های خارجی

- GitHubService
- StatusService
- NotificationService

main_modules 🔌

منطق تجاری اصلی

- ProjectManagementSystem
- TaskManager
- ConfigurationHandler

templates 📄

تولید کد

- HeaderUpdater
- DocumentationGenerator
- TemplateEngine

api 🌐

نقاط پایانی REST API

- برنامه FastAPI
- روترها
- مدل‌های درخواست

🔗 وابستگی‌های سرویس

GitHubService, StatusService, →
ConfigurationService
ProjectManagementSystem

FastAPI, Uvicorn, →
ProjectManagementSystem
API Server

📁 معماری دایرکتوری

/AutoProjectManagement 📁

دایرکتوری اصلی حاوی تمام فایل‌های پروژه

/autoprojectmanagement 📁

/tests ✎

/Docs 📄

/JSonDataBase 🗄

/venv </>

/vscode. ⚙

/autoprojectmanagement 📁

بسته اصلی حاوی کد اصلی برنامه

/main_modules 🔌

/services 🏠

/api 🌐

/templates 📄

/tests ✎

مجموعه تست سازمان‌دهی شده بر اساس دسته

/unit 📋

/integration 🧩

conftest.py ⚙

مدیریت پیکربندی

فایل‌های پیکربندی

pyproject.toml

پیکربندی پروژه و تنظیمات ساخت

```
[build-system]
requires = ["setuptools>=61.0", "wheel"]
"build-backend" = "setuptools.build_meta"
[project]
"name" = "autoprojectmanagement"
"version" = "1.0.0"
"requires-python" = ">=3.8"
```

pytest.ini

پیکربندی چارچوب تست

```
[tool.pytest.ini_options]
testpaths = ["tests"]
python_files = ["test_*.py"]
python_classes = ["Test*"]
python_functions = ["test_*"]
"addopts" = "--cov=autoprojectmanagement"
```

pre-commit-config.yaml

قلاب‌های کیفیت کد و لینتینگ

```
repos:
- repo: https://github.com/psf/black
  rev: 22.3.0
  hooks:
  - id: black
- repo: https://github.com/pycqa/flake8
  rev: 4.0.1
  hooks:
  - id: flake8
```

متغیرهای محیطی

متغیر	توضیح	پیش فرض	الزامی
PYTHONPATH	مسیر جستجوی ماژول	.	بله
ENV	حالت محیط	development	خیر
LOG_LEVEL	سطح لاگ	INFO	خیر
GITHUB_TOKEN	توکن API گیت‌هاب	-	بله*
DATABASE_URL	اتصال پایگاه داده	sqlite:///app.db	خیر

* برای قابلیت‌های یکپارچه‌سازی گیت‌هاب الزامی است

چارچوب تست

اجرای تست‌ها

اجرای همه تست‌ها

اجرای مجموعه تست کامل

```
pytest
```

اجرای دسته خاص

اجرای تست‌ها از یک دسته خاص

```
/pytest tests/unit
```

اجرای با پوشش

تولید گزارش‌های پوشش

```
pytest --cov=autoprojectmanagement --cov-report=html
```

اجرای با نشانگرها

اجرای تست‌ها با نشانگرهای خاص

```
"pytest -m "not slow"
```

اجرای موازی

اجرای تست‌ها به صورت موازی برای اجرای سریع‌تر

```
pytest -n auto
```

معماری تست

تست‌های یکپارچگی

تست تعاملات بین اجزا

مکان: /tests/integration

% پوشش: 80+

تست‌های واحد

تست اجزای جداگانه به صورت ایزوله

مکان: /tests/unit

% پوشش: 90+

تست‌های عملکرد

تست معیارهای عملکرد و حافظه

مکان: /tests/performance

% پوشش: N/A

تست‌های سیستم

تست جریان‌های کاری کامل

مکان: /tests/system

% پوشش: 70+

پیکربندی تست

```
conf/test.py #
@pytest.fixture@
def test_project(tmp_path)
:
    """ایجاد دایرکتوری پروژه تست."""
    project_dir = tmp_path / "test_project"
    project_dir.mkdir()
    return project_dir
```



گردش کار توسعه

فرآیند توسعه

1 توسعه ویژگی

ایجاد شاخه ← توسعه ← نوشتن تست ← اجرای تست

2 بازبینی کد

بررسی‌های خودکار ← بازبینی هم‌تا ← تست‌های یکپارچگی

3 ادغام و استقرار

ادغام با develop ← استقرار در محیط تست ← انتشار

```
git checkout -b feature/new-feature develop
# توسعه ویژگی
git commit -m "feat(api): افزودن نقطه پایانی ایجاد پروژه"
git push origin feature/new-feature
# ایجاد درخواست ادغام به develop
```

گردش کار Git

1 استراتژی شاخه

develop ← main ← شاخه‌های ویژگی/رفع باگ

2 قرارداد Commit

فرمت: نوع(حوزه): توضیح

نوع	فرمت	مثال
feat	feat(حوزه): توضیح	feat(api): افزودن ایجاد پروژه
fix	fix(حوزه): توضیح	fix(git): رفع تداخل ادغام
docs	docs(حوزه): توضیح	docs(readme): به‌روزرسانی نصب

یکپارچه‌سازی مداوم

🔄 راه‌اندازها: Push به هر شاخه، ایجاد درخواست ادغام

☰ ماتریس: تست در چندین نسخه پایتون (3.8، 3.9، 3.10، 3.11)

☰ بررسی‌ها: لینتینگ، بررسی نوع، تست، گزارش پوشش

```
name: CI
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: [3.8, 3.9, 3.10, 3.11]
```


عیب‌یابی

پیکربندی دیباگ

```
version": "0.2.0", "configurations": [ { "name": "Debug" }  
  Auto Runner", "type": "python", "request": "launch",  
"module": "autoprojectmanagement.auto_runner", "args": ["--  
  path", "${workspaceFolder}", "--verbose"], "console":  
"integratedTerminal", "cwd": "${workspaceFolder}", "env":  
  {"PYTHONPATH": "${workspaceFolder}" } }, { "name": "Debug  
  API Server", "type": "python", "request": "launch",  
"module": "autoprojectmanagement.api.main", "args": ["--  
  reload"], "console": "integratedTerminal", "jinja": true }  
  { ]
```

تحلیل لاگ

INFO



auto_project/logs/info.log - عملیات عمومی

WARNING



auto_project/logs/warning.log - مشکلات احتمالی

ERROR



auto_project/logs/error.log - مشکلات بحرانی

مشکلات رایج و راه‌حل‌ها

مشکل	نشانه	راه‌حل
خطاهای وارد کردن	ModuleNotFoundError	بررسی PYTHONPATH، نصب مجدد وابستگی‌ها
تضاد نسخه	هشدار وابستگی	استفاده از محیط مجازی، به‌روزرسانی نیازمندی‌ها
احراز هویت	Unauthorized 401	پیکربندی توکن گیت‌هاب
خطاهای تست	خطای Assertion	بررسی داده‌های تست، به‌روزرسانی Assertion

بهترین روش‌ها

دستورات اساسی

وظیفه	دستور
راه‌اندازی محیط	setup_env.sh/.
شروع توسعه	python -m autoprojectmanagement.auto_runner
اجرای تست‌ها	pytest
قالب‌بندی کد	. black
بررسی نوع	mypy autoprojectmanagement

مکان‌های فایل

منبع	مسیر
کد منبع	/autoprojectmanagement
تست‌ها	/tests
مستندات	/Docs
پیکربندی	pyproject.toml
نیازمندی‌ها	requirements*.txt

منابع پشتیبانی

مشکلات GitHub Issues	بحث‌ها GitHub Discussions
مستندات Project Wiki	مثال‌ها examples/ directory

استانداردهای کیفیت کد

تست ایزوله‌سازی تست‌ها شبیه‌سازی سرویس‌های خارجی هدف پوشش 90%+	سبک Python طول خط: 88 کاراکتر نوع‌نویسی الزامی مستندات سبک گوگل
عملکرد اجرای موازی تست‌ها بارگذاری تنبل عملیات ناهمگام	امنیت به‌روزرسانی منظم وابستگی‌ها اسکن کد متغیرهای محیطی

استانداردهای مستندسازی

کلاس سبک گوگل ویژگی‌ها امضای متدها	ماژول هدف در بالا مثال‌های کاربرد وابستگی‌ها
API مستندات خودکار FastAPI مثال‌های درخواست/پاسخ مستندات اسکیمای	تابع آرگومان‌ها، بازگشت‌ها، خطاها نوع‌نویسی شامل مثال‌های کاربرد