

Лабораторная работа 1. Реализация серверного приложения FastAPI.

Необходимо реализовать полноценное серверное приложение с помощью фреймворка FastAPI с применением дополнительных средств и библиотек.

пра1

Создание конечных точек api и виртуальных баз данных

127.0.0.1:8000/docs

Gmail

翻译

编程课

ISU ИТМО [Интр...

Itmo 体育

Пир5д - Google...

机场

WebLab1 API

1.0.0

OAS 3.1

/openapi.json

实践任务: 基于 FastAPI 实现战士、职业、博客相关接口

default

GET

/warriors_list

获取所有战士列表

GET

/warrior/{warrior_id}

获取单个战士信息

PUT

/warrior/{warrior_id}

更新战士信息

POST

/warrior

创建新战士

DELETE

/warrior/delete/{warrior_id}

删除战士

GET

/professions_list

获取所有职业列表

GET

/profession/{profession_id}

获取单个职业信息

GET

/articles_list

获取所有博客文章列表

GET

/article/{article_id}

获取单个博客文章信息

PUT

/article/{article_id}

更新博客文章信息

POST

/article

创建新博客文章

DELETE

/article/delete/{article_id}

删除博客文章

DELETE

/warrior/delete/{warrior_id}

删除战士

根据 ID 删除战士:

- 找到对应战士则删除, 返回成功消息。
- 未找到则返回 404 错误。

Parameters

Cancel

Name

Description

warrior_id

required

integer

(path)

warrior_id

Execute

Responses

Code

Description

Links

200

Successful Response

No links

Media type

application/json

Content Accept header

Parameters

[Cancel](#)

No parameters

Request body required

application/json

```
{
  "id": 0,
  "race": "string",
  "name": "string",
  "level": 0,
  "profession": {
    "id": 0,
    "title": "string",
    "description": "string"
  },
  "skills": []
}
```

Execute

Responses

Code	Description	Links
200	Successful Response	No links
Media type		
application/json		
Controls Accept header.		
Example Value Schema		
{ "id": 0, "race": "string", "name": "string", "level": 0, "profession": { "id": 0, "title": "string", "description": "string" }, "skills": [] }		

praz

Настройка БД, SQLAlchemy и миграции через Alembic

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/warriors' \
  -H 'accept: application/json'
```

Request URL

http://127.0.0.1:8000/warriors

Server response

Code Details

200

Response body

```
[
  {
    "id": 1,
    "race": "director",
    "name": "Мартьянов Дмитрий",
    "level": 12,
    "skills": [
      {
        "id": 1,
        "name": "Купле-продажа компрессоров",
        "description": ""
      },
      {
        "id": 2,
        "name": "Оценка имущества",
        "description": ""
      }
    ]
  },
  {
    "id": 2,
    "race": "worker",
    "name": "Андрей Косякин",
    "level": 12,
    "skills": []
  }
]
```

Response headers

```
content-length: 329
content-type: application/json
date: Sat, 07 Jun 2025 23:48:41 GMT
server: uvicorn
```

Responses

Code Description Links

GET

/warrior/{warrior_id} 获取单个战士信息

^

Parameters

Cancel

Name	Description
warrior_id ★ required integer (path)	<input type="text" value="warrior_id"/>

Execute

Responses

Code	Description	Links
200	Successful Response	No links
<div>Media type <input type="text" value="application/json"/></div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>{ "id": 0, "race": "string", "name": "string", "level": 0, "skills": []}</pre></div>		
422	Validation Error	No links
<div>Media type <input type="text" value="application/json"/></div> <div>Example Value Schema</div> <div><pre>{ "detail": [{ "loc": [</pre></div>		

GET

/users_by_warrior/{warrior_id} 根据战士 ID 获取喜欢该战士的用户

^

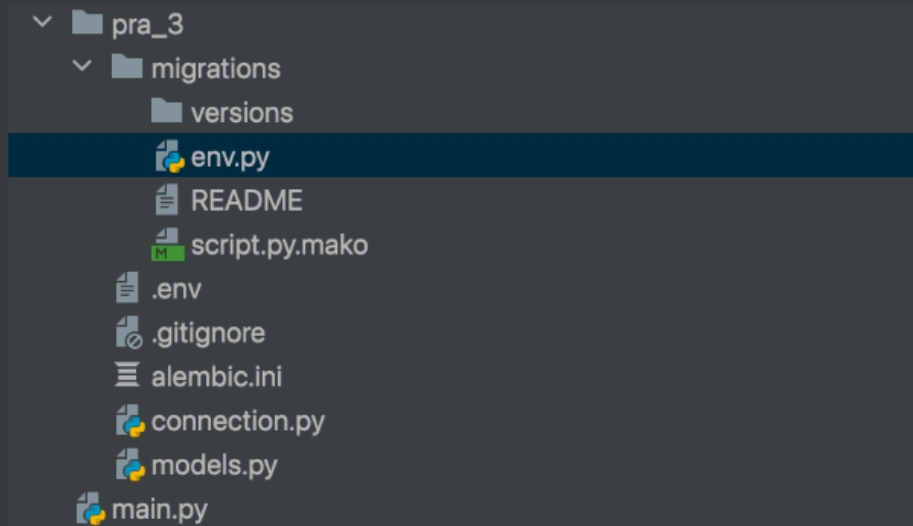
Parameters

Try it out

Name	Description
warrior_id ★ required integer (path)	<input type="text" value="warrior_id"/>

Responses

Code	Description	Links
200	Successful Response	No links
<div>Media type <input type="text" value="application/json"/></div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>[{ "id": 0, "name": "string", "email": "string", "password": "string", "favorite_warriors": [] }]</pre></div>		
422	Validation Error	No links
<div>Media type <input type="text" value="application/json"/></div> <div>Example Value Schema</div> <div><pre>{ "detail": [{</pre></div>		



Лабораторная работа 1

No parameters

Request body required

application/json

```

{
  "id": 0,
  "username": "zining",
  "email": "6666@gmail.com",
  "password": "123456"
}

```

Execute

Clear

Responses

Curl

```

curl -X 'POST' \
  'http://0.0.0.0:8000/api/v1/users/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "username": "zining",
    "email": "6666@gmail.com",
    "password": "123456"
  }'

```

Request URL

```

http://0.0.0.0:8000/api/v1/users/

```

Server response

Code

Details

No parameters

Request body required

application/json

```

{
  "id": 0,
  "amount": 10000,
  "description": "",
  "transaction_type": "expense",
  "created_at": "2025-06-08T01:45:51.611Z",
  "user_id": 0,
  "category_id": 0
}

```

Responses

Curl

```
curl -X 'POST' \
  'http://0.0.0.0:8000/api/v1/transactions/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "amount": 10000,
    "description": "",
    "transaction_type": "expense",
    "created_at": "2025-06-08T01:45:51.611Z",
    "user_id": 0,
    "category_id": 0
  }'
```

Request URL

http://0.0.0.0:8000/api/v1/transactions/

Request body required

application/json

```
{
  "id": 0,
  "category": "string",
  "amount": 100000,
  "start_date": "2025-06-08T01:46:45.468Z",
  "end_date": "2025-06-08T01:46:45.468Z",
  "user_id": 0
}
```

28

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://0.0.0.0:8000/api/v1/budgets/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "category": "string",
    "amount": 100000,
    "start_date": "2025-06-08T01:46:45.468Z",
    "end_date": "2025-06-08T01:46:45.468Z",
    "user_id": 0
  }'
```

Request URL

http://0.0.0.0:8000/api/v1/budgets/

Navicat Premium

连接 新建查询 表 视图 函数 用户 其它 查询 备份 自动运行 模型 图表 查看

对象

名	行	数据长度	引擎	创建日期	修改日期
budget	0	16 KB	InnoDB	2025-06-08 04:43:39	
budgetexpenselink	0	16 KB	InnoDB	2025-06-08 04:43:39	
category	0	16 KB	InnoDB	2025-06-08 04:43:39	
financialgoal	0	16 KB	InnoDB	2025-06-08 04:43:39	
transaction	0	16 KB	InnoDB	2025-06-08 04:43:39	
user	0	16 KB	InnoDB	2025-06-08 04:43:39	

finance_db 已打开

Wzn

字符集 utf8mb4

排序规则 utf8mb4_unicode_ci

POST

/api/v1/goals/ Create Goal

Parameters

No parameters

Request body required

application/json

```
{
  "id": 0,
  "title": "string",
  "target_amount": 800000,
  "current_amount": 10000,
  "deadline": "2025-06-08T01:49:40.635Z",
  "is_achieved": false,
  "user_id": 0
}
```

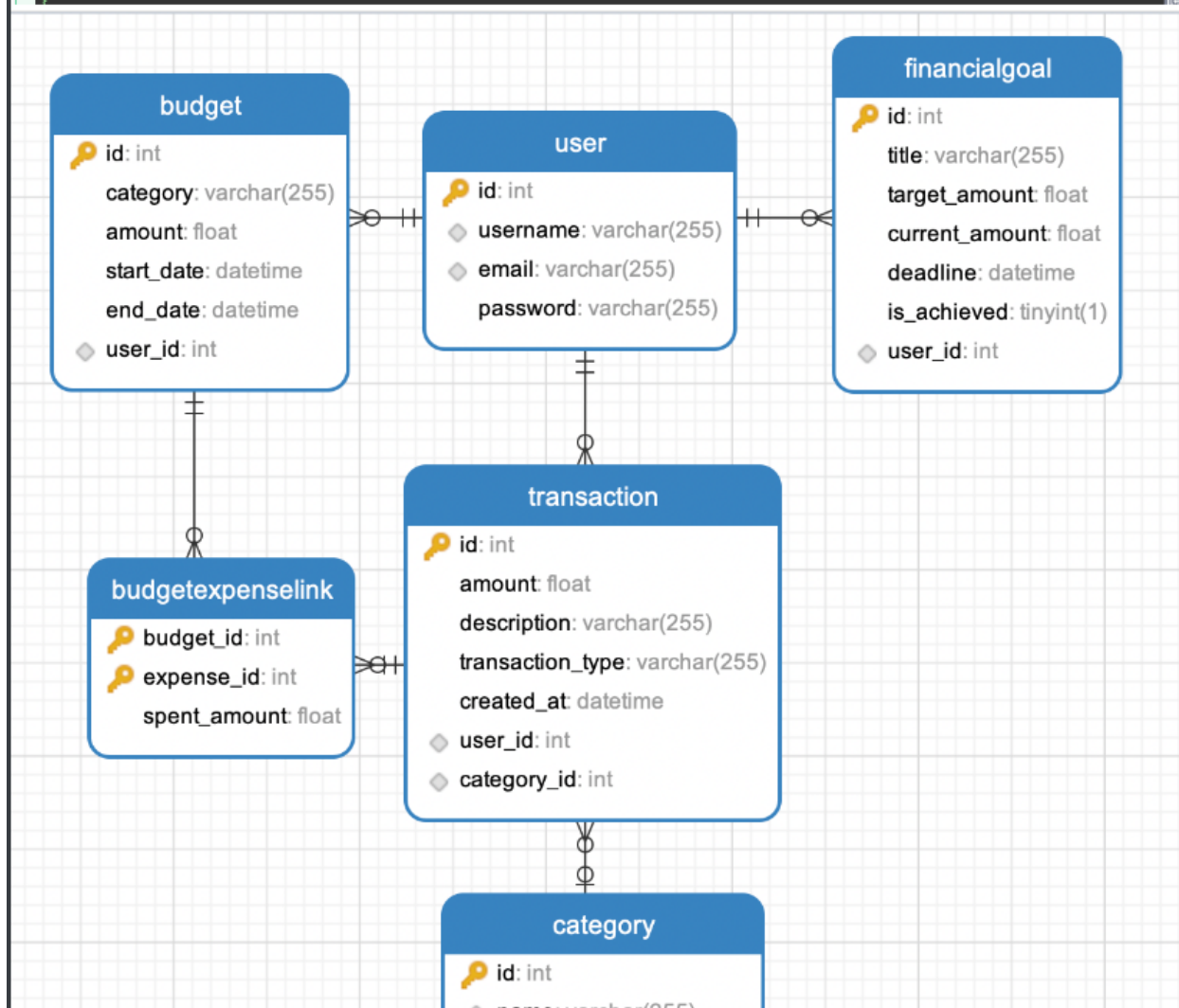
Execute

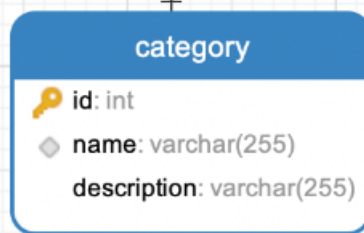
Clear

Responses

Curl

```
curl -X 'POST' \
  'http://0.0.0.0:8000/api/v1/goals/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "title": "string",
    "target_amount": 800000,
    "current_amount": 10000,
    "deadline": "2025-06-08T01:49:40.635Z",
    "is_achieved": false,
    "user_id": 0
  }'
```





Лабораторная работа 2. Потоки. Процессы. Асинхронность.

Цель работы: понять отличия потоками и процессами и понять, что такое асинхронность в Python.

Работа о потоках, процессах и асинхронности поможет студентам развить навыки создания эффективных и быстродействующих программ, что важно для работы с большими объемами данных и выполнения вычислений.

Задача 1. Различия между `threading`, `multiprocessing` и `async` в Python

Задача: Напишите три различных программы на Python, использующие каждый из подходов: `threading`, `multiprocessing` и `async`. Каждая программа должна решать считать сумму всех чисел от 1 до 10000000000000. Разделите вычисления на несколько параллельных задач для ускорения выполнения.

`threading.py`:

```
Run: threading_sum x
/Users/not_hungry_king/web_lab2/.venv/bin/python /Users/not_hungry_king/web_lab2/task1/threading_sum.py
Threading sum = 49999995000000
Time: 0.19 seconds
Process finished with exit code 0
```

`Multiprocessing` — Процессы, `multiprocessing.py`:

```
Run: multiprocessing_sum x
/Users/not_hungry_king/web_lab2/.venv/bin/python /Users/not_hungry_king/web_lab2/task1/multiprocessing_sum.py
Multiprocessing sum = 49999995000000
Time: 0.11 seconds
Process finished with exit code 0
```

`Async` - Асинхронность, `async.py`:

```
Run: async_sum x
/Users/not_hungry_king/web_lab2/.venv/bin/python /Users/not_hungry_king/web_lab2/task1/async_sum.py
AsyncIO sum = 49999995000000
Time: 0.18 seconds
Process finished with exit code 0
```

Задача 2. Параллельный парсинг веб-страниц с сохранением в базу данных

Задача: Напишите программу на Python для параллельного парсинга нескольких веб-страниц с сохранением данных в базу данных с использованием подходов `threading`, `multiprocessing` и `async`. Каждая программа должна парсить информацию с нескольких веб-сайтов, сохранять их в базу данных.

`threading.py`:

```
Terminal:  Local × Local (2) × + ▾
Threading 完成: https://www.example.com → Example Domain
Threading 完成: https://pydantic.dev → Pydantic

Threading 完成: https://sqlalchemy.org →

SQLAlchemy - The Database Toolkit for Python

=== Threading 结果 ===
总耗时: 20.63 秒
数据库表: web_pages (查看 finance_db 库验证数据)
(.venv) not_hungry_king@baodawangdeMacBook-Air task2 %
```

Multiprocessing — Процессы, `multiprocessing.py`:

```
Run:  task2_multiprocessing × multiprocessing_sum ×
warnings.warn(
Multiprocessing 完成: https://www.example.com → Example Domain
Multiprocessing 完成: https://www.python.org → Welcome to Python.org
Multiprocessing 完成: https://fastapi.tiangolo.com → FastAPI
Multiprocessing 完成: https://pydantic.dev → Pydantic
Multiprocessing 完成: https://sqlalchemy.org →

SQLAlchemy - The Database Toolkit for Python

=== Multiprocessing 结果 ===
总耗时: 2.32 秒
数据库表: web_pages (查看 finance_db 库验证数据)
```


Async - Асинхронность, async.py:

```
Run: task2_async x multiprocessing_sum x
Base = declarative_base()
Async 完成: https://www.example.com → Example Domain
Async 完成: https://fastapi.tiangolo.com → FastAPI
Async 完成: https://www.python.org → Welcome to Python.org
Async 完成: https://pydantic.dev → Pydantic
Async 完成: https://sqlalchemy.org →

SQLAlchemy - The Database Toolkit for Python

=== Async 结果 ===
总耗时: 1.03 秒
数据库表: web_pages (查看 finance_db 库验证数据)

Process finished with exit code 0
```

Лабораторная работа 3: Упаковка FastAPI приложения в Docker, Работа с источниками данных и Очереди

Цель: Научиться упаковывать FastAPI приложение в Docker, интегрировать парсер данных с базой данных и вызывать парсер через API и очередь.

Lab3 FastAPI App 1.0.0 OAS 3.1

/openapi.json

Docker + 队列 + 数据源实验

tasks

POST /tasks/tasks/submit 提交任务到队列

GET /tasks/tasks/{task_id} 查询任务状态

Schemas

HTTPValidationError > Expand all object

ValidationError > Expand all object

код из файла Dockerfile:

```
FROM python:3.9-slim
WORKDIR /app
COPY Requirements.txt .
RUN pip install --no-cache-dir -r Requirements.txt
COPY . .
EXPOSE 8000
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

код из файла Docker-compose.yml

```
version: "3.8"

services:
  web:
    build:
      context: .
      dockerfile: Docker/Dockerfile
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=mysql+pymysql://root:wzn001021@db/finance_db
      - REDIS_URL=redis://redis:6379/0
    depends_on:
      - db
      - redis

  worker:
    build:
      context: .
      dockerfile: Docker/Dockerfile
    command: celery -A app.celery_config.celery_app worker --loglevel=info
    environment:
      - DATABASE_URL=mysql+pymysql://root:wzn001021@db/finance_db
      - REDIS_URL=redis://redis:6379/0
    depends_on:
      - db
      - redis

  db:
    image: mysql:8.0
    environment:
      - MYSQL_ROOT_PASSWORD=wzn001021
      - MYSQL_DATABASE=finance_db
    volumes:
      - mysql_data:/var/lib/mysql

  redis:
    image: redis:7.0-alpine
    ports:
      - "6379:6379"

volumes:
  mysql_data:
```

код из файла celery_config.py:

```
from celery import Celery

from os import getenv

celery_app = Celery(
    "lab3_worker",
    broker=getenv("REDIS_URL", "redis://localhost:6379/0"),
    backend=getenv("REDIS_URL", "redis://localhost:6379/0"),
    include=["app.routers.tasks"]
)

celery_app.conf.timezone = "UTC"
celery_app.conf.task_serializer = "json"
```

Запуск бэкэнда celery и клиента FASTapi:

```
Terminal: Local + -
=> [web 5/5] COPY . . 8.8s
=> [web] exporting to image 4.0s
=> => exporting layers 2.0s
=> => exporting manifest sha256:81b811cf919c786402ef3ab900dc407a3020850825ad9dfb57357e86d4f87e9 8.0s
=> => exporting config sha256:c7be48eb6aca182c8cde19746e521d4c9f3f2af0cbb3ab288e419ef6fa773f83 8.0s
=> => exporting attestation manifest sha256:f8ae55857e4399dec3aed684b0c8f5b3486efa3efe8f1bbc0b7796cb49c1e5cd 8.0s
=> => exporting manifest list sha256:e8ad3936a026fe42c22e5478d50se3e9b53a3a8ddcf3961ceb9da842e9b953c1c 8.0s
=> => naming to docker.io/library/web_lab3-web:latest 6.0s
=> => unpacking to docker.io/library/web_lab3-web:latest 1.4s
[web] resolving provenance for statdata file 8.1s
[+] Building 1/1
  ✓ web Built 8.8s
  ...
(.venv) not_hungry_king@baodexangdeMacBook-Air web_lab3 % docker-compose up -d
WARN[0808] /Users/not_hungry_king/web_lab3/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 21/21
  ✓ redis Pulled 6.0s
  ✓ db Pulled 29.6s
[+] Running 5/5
  ✓ Network web_lab3_default Created 8.1s
  ✓ Volume "web_lab3_db_data" Created 6.0s
  ✓ Container web_lab3-redis-1 Started 8.7s
  ✓ Container web_lab3-db-1 Started 6.7s
  ✓ Container web_lab3-web-1 Started 8.7s
```

Выполните функциональное тестирование:

Name	Description
task_data required	test
string	
(query)	

Execute

Clear

Responses

Curl

curl -X 'POST' \
 'http://0.0.0.0:8000/tasks/submit?task_data=test' \
 -H 'accept: application/json' \
 -d ''

Request URL

http://0.0.0.0:8000/tasks/submit?task_data=test

Server response

Code	Details
200	<div>Response body<div>{ "task_id": 1, "status": "queued", "message": "任务已提交到队列，异步处理中"}</div><div>Download</div></div> <div>Response headers<div>content-length: 86 content-type: application/json date: Sun, 08 Jun 2025 20:31:05 GMT server: uvicorn</div></div>

Responses

Parameters

Cancel

Response headers

```
content-length: 86
content-type: application/json
date: Sun, 08 Jun 2025 20:31:05 GMT
server: uvicorn
```

Responses
Parameters

Cancel

NameDescription

task_data * required
string
(query)

testno1

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://0.0.0.0:8000/tasks/submit?task_data=testno1' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

http://0.0.0.0:8000/tasks/submit?task_data=testno1

Server response

CodeDetails

200

Response body

```
{
  "task_id": 1,
  "status": "queued",
  "message": "任务已提交到队列，异步处理中"
}
```



Download

Response headers

```
content-length: 86
content-type: application/json
date: Sun, 08 Jun 2025 20:41:49 GMT
server: uvicorn
```

Responses

CodeDescription

Links

NameDescription

task_id * required
integer
(path)

1

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://0.0.0.0:8000/tasks/1' \
  -H 'accept: application/json'
```

Request URL

<http://0.0.0.0:8000/tasks/1>

Server response

CodeDetails

200

Response body

```
{
  "task_id": 1,
  "status": "completed",
  "data": "testno1",
  "created_at": "2025-06-08T23:41:50",
  "completed_at": "2025-06-08T23:41:56"
}
```



Download

Response headers

```
content-length: 123
content-type: application/json
date: Sun, 08 Jun 2025 20:42:25 GMT
server: uvicorn
```

Responses

CodeDescription

Links

Responses

Code

Description

Links

База данных бэк-офиса:

django_demo

Wzn

django_demo

finance_db

表

budget

budgetexpenselink

category

financialgoal

lab3_tasks

transaction

user

web_pages

视图

函数

事件

查询

备份

information_schema

mysql

performance_schema

personal_finance

sys

website

对象

lab3_tasks@finance_db...

id

status

data

description

created_at

completed_at

1

completed

testno1

Lab3 队列任务

2025-06-08 23:41:50

2025-06-08 23:41:56

lab3_tasks

表

Wzn

finance_db

行

1

数据长度

16 KB (16,384)

引擎

InnoDB

创建日期

2025-06-08 23:41:25

修改日期

2025-06-08 23:41:55

排序规则

utf8mb4_unicode_ci

行格式

Dynamic

平均行长度

0 B (0)

最大数据长度

0 B (0)

索引长度

0 B (0)

检查时间

--

自动递增

2

SELECT * FROM 'finance_db'. 'lab3_...

1 条记录在第 1 页

Терминал выполнения и сельдерейный дисплей:

Terminal: Local x Local [2] + v

[tasks]
 tasks.process_task

[2025-06-08 23:36:41,436: INFO/MainProcess] Connected to redis://localhost:6379/0
[2025-06-08 23:36:41,445: INFO/MainProcess] mingle: searching for neighbors
[2025-06-08 23:36:42,470: INFO/MainProcess] mingle: all alone
[2025-06-08 23:36:42,554: INFO/MainProcess] celery@baodawangdeMacBook-Air.local ready.
[2025-06-08 23:37:20,122: INFO/MainProcess] Task tasks.process_task[baleee82-06a7-4946-a527-d56774cb708e] received
[2025-06-08 23:37:25,549: INFO/ForkPoolWorker-7] Task tasks.process_task[baleee82-06a7-4946-a527-d56774cb708e] succeeded in 5.311484458000000s: None
[2025-06-08 23:41:50,373: INFO/MainProcess] Task tasks.process_task[ee579c8-e396-4414-b7f2-94ad7ffd5595] received
[2025-06-08 23:41:55,720: INFO/ForkPoolWorker-7] Task tasks.process_task[ee579c8-e396-4414-b7f2-94ad7ffd5595] succeeded in 5.266497787999974s: None

III TOOO Problems Terminal Python Packages Pytl App Store
Packages installed successfully: installed packages: 'python-dotenv==0.20.0' (22 minutes ago)

35:38 LF UTF-8 4 spaces Python

not_hungry_king@baodawangdeMacBook-Air web_lab3 % uvicorn app.main:app --host 0.0.0.0 --port 8000

✓数据库连接正常
✓数据库连接正常
INFO: Started server process [20184]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO: 127.0.0.1:52685 - "GET / HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:52701 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:52701 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:52714 - "POST /tasks/submit?task_data=testno1 HTTP/1.1" 200 OK
INFO: 127.0.0.1:52743 - "GET /tasks/1 HTTP/1.1" 200 OK