Course FastAPI

Table of Content

[1 REPO 2](#_Toc152109806)

[2 Lesson0 Environment: 2](#_Toc152109807)

[3 Lesson1 Application, documentation 2](#_Toc152109808)

[4 Lesson2a 4](#_Toc152109809)

[4.1 New app 4](#_Toc152109810)

[5 Lesson2 Эндпоинты, Параметры URL и Запроса 6](#_Toc152109811)

[5.2 Get Request with 2 parameters 6](#_Toc152109812)

[5.3 Post query: change name of user: 8](#_Toc152109813)

[6 Lesson3 FastAPI - Валидация данных с Pydantic #3 8](#_Toc152109814)

[6.2 New class + datamodel 9](#_Toc152109815)

[6.3 Run uvicorn main:app –reload 9](#_Toc152109816)

[6.4 Полная структура данных 13](#_Toc152109817)

[6.5 If need that Customer will see validation Error: 16](#_Toc152109818)

[7 Lesson #8 Caching with Redis 17](#_Toc152109819)

[7.1 Environment 17](#_Toc152109820)

[7.2 Cache in FastAPI with Redis 18](#_Toc152109821)

[7.3 Environment pip install fastapi-cache2[redis] 18](#_Toc152109822)

[7.4 Run app 18](#_Toc152109823)

[7.5 Test cache : 18](#_Toc152109824)

[8 Lesson #9 Фоновые задачи c Celery Redis Flower 20](#_Toc152109825)

[8.1 Регистрация приложения в Google: 2-факт аутентификация – username – password 20](#_Toc152109826)

[8.2 Environment : Celery, Flower 20](#_Toc152109827)

[8.3 Инициализация Celery : 20](#_Toc152109828)

[8.4 run celery in new terminal 21](#_Toc152109829)

[9 Lesson10 23](#_Toc152109830)

|  |
| --- |
| REPO <https://github.com/ecrvmal/Trading_app>  <https://github.com/artemonsh/fastapi_course> |
| Lesson0 Environment: <https://www.youtube.com/watch?v=7IdfnjXsdN4&list=PLeLN0qH0-mCVQKZ8-W1LhxDcVlWtTALCS> |
| Install FastAPIpip install fastapi[all]applications: pydantic – serializer, verification  starlette : base of FastAPI  uvicorne : web server; |
| Lesson1 Application, documentation <https://www.youtube.com/watch?v=G0pcbxMsiec&list=PLeLN0qH0-mCVQKZ8-W1LhxDcVlWtTALCS&index=2> First app:Edit main.py from fastapi import FastAPI  app = FastAPI()  @app.get("/") def hello():  return "Hello world" |
| Run app: (venv) PS D:\GB\pythonProject\Trading\_app> uvicorn main:app --reload  INFO: Will watch for changes in these directories: ['D:\\GB\\pythonProject\\Trading\_app']  INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)  INFO: Started reloader process [13240] using WatchFiles  INFO: Started server process [12688]  INFO: Waiting for application startup.  INFO: Application startup complete.   file:app\_name autoreload |
| Docs 1:<http://127.0.0.1:8000/docs>   Hello == function name |
|  |
| Docs 2:<http://127.0.0.1:8000/redoc> |
| Lesson2aNew app from fastapi import FastAPI  app = FastAPI(  title="Training App")  fake\_users =[  {"id": 1, 'role': "admin" , 'name': 'Bob'},  {"id": 2, 'role': "investor", 'name': 'John'},  {"id": 3, 'role': "trader", 'name': 'Matt'}, ]  @app.get("/users/{user\_id}") def get\_user(user\_id):  return user\_id  > uvicorn main:app --reload  <http://127.0.0.1:8000/users/2>  “2”  <http://127.0.0.1:8000/docs> |
| Lesson2 Эндпоинты, Параметры URL и ЗапросаEdit src/main.py from fastapi import FastAPI  app = FastAPI(  title="Trading App" )  fake\_users = [  {"id": 1, "role": "admin", "name": "Bob"},  {"id": 2, "role": "investor", "name": "John"},  {"id": 3, "role": "trader", "name": "Matt"}, ]  @app.get("/users/{user\_id}") def get\_user(user\_id: int):  return [user for user in fake\_users if user.get("id") == user\_id] test1 uvicorn main:app –reload  <http://127.0.0.1:8000/users/2>  [{"id":2,"role":"investor","name":"John"}]  <http://127.0.0.1:8000/docs> |
| Get Request with 2 parametersEdit main.py fake\_trades = [  {"id": 1, "user\_id": 1, "currency": "BTC", "side": "buy", "price": 123, "amount": 2.12},  {"id": 2, "user\_id": 1, "currency": "BTC", "side": "sell", "price": 125, "amount": 2.12}, ]   @app.get("/trades") def get\_trades(limit: int = 1, offset: int = 0):  return fake\_trades[offset:][:limit]  <http://127.0.0.1:8000/trades?limit=1&offset=1>  [{"id":2,"user\_id":1,"currency":"BTC","side":"sell","price":125,"amount":2.12}] |
| Post query: change name of user:Edit main.py fake\_users2 = [  {"id": 1, "role": "admin", "name": "Bob"},  {"id": 2, "role": "investor", "name": "John"},  {"id": 3, "role": "trader", "name": "Matt"}, ]   @app.post("/users/{user\_id}") def change\_user\_name(user\_id: int, new\_name: str):  current\_user = list(filter(lambda user: user.get("id") == user\_id, fake\_users2))[0]  current\_user["name"] = new\_name  return {"status": 200, "data": current\_user}  uvicorn main:app –reload  Browser: http://127.0.0.1:8000/docs  request POST  **http://127.0.0.1:8000/users/1?new\_name=Jessica**  Responce:   | **Code** | **Details** | | --- | --- | | 200 | Response body **Download**  **{**  **"status": 200,**  **"data": {**  **"id": 1,**  **"role": "admin",**  **"name": "Jessica"**  **}**  **}** | |
| Lesson3 FastAPI - Валидация данных с Pydantic #3Environment Pip install pydentic |
| New class + datamodel Edit main.py  # to describe Trade we use data model (Pydentic) class Trade(BaseModel):  id: int  user\_id: int  currency: str  side: str  price: float  amount: int   @app.post("/trades") def add\_trades(trades: List[Trade]): # input : list of deals  fake\_trades.extend(trades)  return {"status": 200, "data": fake\_trades} |
| Run uvicorn main:app –reload Browser: 127.0.0.1:8000/docs Response template POST End-point появился, параметры не появились  Структура данных понятна из класса Trade |
| Если поле не соответетвует типу в datamodel - будет ошибка |
| Check input data: from pydantic import BaseModel, Field  # to describe Trade we use data model (Pydentic) class Trade(BaseModel):  id: int  user\_id: int  currency: str = Field(max\_length=5)  side: str  price: float = Field(ge=0)  amount: int |
| Checking of sending data from pydentic Edit app/main.py  class User(BaseModel):  id: int  role: str  name: str  @app.get("/users/{user\_id}", response\_model=List[User]) def get\_user(user\_id: int):  return [user for user in fake\_users if user.get("id") == user\_id]  run browser:  127.0.0.1/docs    ... |
| Видим структуру данных ответа – реагирует на ошибки, если валидация не прошла, данные не возвращает.    Клиент не видит какая ошибка произошла на back-end (м.б. уязвимость) |

|  |
| --- |
| Полная структура данныхEdit main.py from datetime import datetime from typing import List, Optional  from fastapi import FastAPI from pydantic import BaseModel, Field  app = FastAPI(  title="Trading App" )  fake\_users = [  {"id": 1, "role": "admin", "name": "Bob"},  {"id": 2, "role": "investor", "name": "John"},  {"id": 3, "role": "trader", "name": "Matt"}, ]  class DegreeType(BaseModel):  newbie = "newbie"  expert = "expert"  class Degree(BaseModel):  id: int  created\_at: datetime  type\_degree: DegreeType # newbie | expert if else > error  # see from class DegreeType  class User(BaseModel):  id: int  role: str  name: str # degree: Optional[list[Degree]] # either list or null  # degree: Optional[list[Degree]]=[] # either list or default []  degree: list[Degree] # see model from class Degree  # response\_model from class User above @app.get("/users/{user\_id}", response\_model=List[User]) def get\_user(user\_id: int):  return [user for user in fake\_users if user.get("id") == user\_id]  # request : http://127.0.0.1:8000/users/2 # [{"id":2,"role":"investor","name":"John"}]   fake\_trades = [  {"id": 1, "user\_id": 1, "currency": "BTC", "side": "buy", "price": 123, "amount": 2.12},  {"id": 2, "user\_id": 1, "currency": "BTC", "side": "sell", "price": 125, "amount": 2.12}, ] |
| @app.get("/trades") def get\_trades(limit: int = 1, offset: int = 0):  return fake\_trades[offset:][:limit]  # http://127.0.0.1:8000/trades?limit=1 # [{"id":1,"user\_id":1,"currency":"BTC","side":"buy","price":123,"amount":2.12}] # http://127.0.0.1:8000/trades?limit=1&offset=1 # [{"id":2,"user\_id":1,"currency":"BTC","side":"sell","price":125,"amount":2.12}]       # to describe Trade we use data model (Pydentic) class Trade(BaseModel):  id: int  user\_id: int  currency: str = Field(max\_length=5)  side: str  price: float = Field(ge=0)  amount: int   @app.post("/trades") def add\_trades(trades: List[Trade]): # input : list of deals  fake\_trades.extend(trades)  return {"status": 200, "data": fake\_trades} |
| Response template on GET: Browser: 127.0.0.1:8000/docs |
| If need that Customer will see validation Error: If validation Error happens:  # Благодаря этой функции клиент видит ошибки, происходящие на сервере, # вместо "Internal server error" @app.exception\_handler(ValidationError) async def validation\_exception\_handler(request: Request, exc: ValidationError):  return JSONResponse(  status\_code=status.HTTP\_422\_UNPROCESSABLE\_ENTITY,  content=jsonable\_encoder({"detail": exc.errors()}), # client will see error detail  )  It will be intercepted by validation\_exception\_handler,  In case validation Error, Customer will see: (GET example) |

|  |
| --- |
| Lesson4 Базы данных и миграции Alembic |
| Install postgress for windows Скачать PostgreSQL для Windows: [https://www.enterprisedb.com/download...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbm5YSkZpbnowenQwSWRCWWFCTDlJeDlfNmFUZ3xBQ3Jtc0tuYVZpaEc3TkhKOHZwdW1pYmtBSkZnX0phT0pGUVpUalBjZ25jdWZJdl9BMkk5aFhSNXF6MWxpcGtmdU1Ra2ZpUldBa1BSaHNvdjB4V3RLbnJQLVpCRExrNVFTVmpiT3lBTGphY2ZGSHZPeXNoR293bw&q=https%3A%2F%2Fwww.enterprisedb.com%2Fdownloads%2Fpostgres-postgresql-downloads&v=hO7b4yh-Qfs)  Скачать PostgreSQL для Linux/MacOS: [https://www.digitalocean.com/communit...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbGQ1OVUxYkNBS0EtcGR0Yi1QdXJETFNMWEFOZ3xBQ3Jtc0ttWHZjQjVQQ25hNFlLT3VmWTVmZ0dZRWJPNmFzbzRuMzhYMklJRDlHbks0dHVTOXZiVTFWYURHQkxCNFFZTGJJSFJvTjBOQ05ocUozWDR0ODNHYnR6UTR5SjBFVXI5WDIzQmNrd2hBc3ZUNWZIdU9xUQ&q=https%3A%2F%2Fwww.digitalocean.com%2Fcommunity%2Ftutorials%2Fhow-to-install-and-use-postgresql-on-ubuntu-18-04&v=hO7b4yh-Qfs)  Скачать pgAdmin для Windows: [https://www.pgadmin.org/download/pgad...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbG5xbXlMNUt6c0hZWTUxZjU5cnI0eUF2cnVqQXxBQ3Jtc0tuanV4TVZwNjN0ckJyZ1RzZElVaXB1NFZBYUpkZlFlRzZRa0o0VFNHb2VuU0VVYVNqNU1yVW1RaXd0c0YwQ0FoamxNYVVzT0dmSktqdG9SS2NoLUtBRlVXSGpjNTk0UV9nVEZjLTVxUmE3ZjNiVG12dw&q=https%3A%2F%2Fwww.pgadmin.org%2Fdownload%2Fpgadmin-4-windows%2F&v=hO7b4yh-Qfs)  Скачать pgAdmin для Linux/MacOS: [https://www.pgadmin.org/download/pgad...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbTZ5YWRwb2hLYWxkV3hYRFVjVDlvZVljb3FxUXxBQ3Jtc0ttN3VYdmpQbUI0RkdjNmtyR095OHktUFdFYlZSQVVzM2xLaXV6eWdqLWh0X0VvVEx2cDkxTjFheEZMdElxYUVvdnRFMEVlZEFUV3ZsSUFwTXhxUXpwWGJjVGNreGFteXNSczB2dGs0NWJXdW8wNjNlcw&q=https%3A%2F%2Fwww.pgadmin.org%2Fdownload%2Fpgadmin-4-apt%2F&v=hO7b4yh-Qfs)  В PGAdmin:   * Enter password * Servers > Create Server > Name * Host : Localhost * Password “postgre” |
| Environment Pip install SQL Alchemy pip install alembic - for migrations  Connector for BD:  pip install psycopg2-bynary |
| ModelsEdit proj/models/models.py |
|  |

|  |
| --- |
| Lesson #8 Caching with Redis |
| EnvironmentInstall Redis for Windows : https://github.com/tporadowski/redis/releases |
| Download zip  Unpack zip  redis-server.exe # run on widows  redis-cli.exe # run on widows |
| redis-cli.exe |
| Cache in FastAPI with Redis Environment:  <http://github.com/long2ice/fastapi-cache> > source + readme  <http://github1s.com/long2ice/fastapi-cache> > # format of VSCode |
| Environment pip install fastapi-cache2[redis]Edit main/py @app.on\_event("startup") async def startup\_event():  redis = aioredis.from\_url("redis://localhost", encoding="utf8",   decode\_responses=True)  FastAPICache.init(RedisBackend(redis), prefix="fastapi-cache")  Startup # действия при старте uvicorn  Shutdown # действия при окончании uvicorn  FastAPICache - инициация класса  # cache example @app.get("/") @cache(expire=60) async def index():  return dict(hello="world") |
| Run app $ uvicorn main:app –reload |
| Test cache :Edit src/operations/router/py ...  @router.get("/long\_operation")  @cache(expire=30)  def get\_long\_op():  time.sleep(2)  return "Много много данных, которые вычислялись сто лет" run $ uvicorn main:app –reload  <http://127.0.0.1:8000/docs>   * Long operations > execute ( 1-й раз долго думает из-за sleep(2) ) * Long operations > execute ( 2-й раз мгновенно – из cache )     Check cache |

|  |
| --- |
| Lesson #9 Фоновые задачи c Celery Redis Flower |
| Регистрация приложения в Google: 2-факт аутентификация – username – passwordEnvironment : Celery, Flower Pip install celery фоновые задачи  Pip install flower |
| Инициализация Celery : |
| Edit src/tasks/tasks.py import smtplib from email.message import EmailMessage from celery import Celery gfrom config import SMTP\_USER, SMTP\_PASSWORD  SMTP\_HOST : ”smtp.gmail.com“ SMTP\_PORT = 465  celery = Celery('tasks', broker='redis://localhost:6379') # fastapi app  def get\_email\_template\_dashboard(username: str):  email = EmailMessage()  email['Subject'] = 'Information for '  email['From'] = SMTP\_USER  email['To'] = SMTP\_USER  email.set\_content(  '<div>'  f'<h1 style=”color: Ped;">Hello {username}, here is your report, please check </h1>' \  '<img src=" https://sun9-18.userapi.com/impf/c624316/v624316886/399de/jkz4qhE0MVw.jpg" ' \  'style="width:600 px; height: 600px;”>'  '</div>',  subtype = 'html'  )  return email   @celery.task # fastAPI decorator for celery app def send\_email\_report\_dashboard(username: str):  email = get\_email\_template\_dashboard(username)  with smtplib.SMTP\_SSL(SMTP\_HOST, SMTP\_PORT) as server:  server.login(SMTP\_USER, SMTP\_PASSWORD)  server.send\_message(email) |
| Edit src/tasks/router.py import  from tasks.tasks import send\_email\_report\_dashboard  router = APIRouter(prefix=”/report”)   @router.get(”/dashboard”) def get\_dashboard\_report(user=Depends(current\_user)): # fastAPI Authenticated user  send\_email\_report\_dashboard(user.username)  return {”status”: 200, ”data”: ”mail sent”, ”details”: None} run celery in new terminal windows:  Trading\_app> celery -A tasks.tasks:celery worker --loglevel=INFO --pool=solo    Linux:  $ Trading\_app> celery -A tasks.tasks:celery worker --loglevel=INFO  -A расположение app\_celery  Задания:  Worker  Flower  bit |
| (venv) app> celery -A tasks.tasks:celery worker --loglevel=INFO --pool=solo    -------------- celery@DESKTOP-ULH9S6I v5.3.6 (emerald-rush)  --- \*\*\*\*\* -----  -- \*\*\*\*\*\*\* ---- Windows-10-10.0.19045-SP0 2023-11-25 23:02:56  - \*\*\* --- \* ---  - \*\* ---------- [config]  - \*\* ---------- .> app: tasks:0x1975a3f7010  - \*\* ---------- .> transport: redis://localhost:6379//  - \*\* ---------- .> results: disabled://  - \*\*\* --- \* --- .> concurrency: 8 (solo)  -- \*\*\*\*\*\*\* ---- .> task events: OFF (enable -E to monitor tasks in this worker)  --- \*\*\*\*\* -----  -------------- [queues]  .> celery exchange=celery(direct) key=celery  [tasks]  . tasks.tasks.send\_email\_report\_dashboard.......  [2023-11-25 23:02:56,251: INFO/MainProcess] mingle: searching for neighbors  [2023-11-25 23:02:57,271: INFO/MainProcess] mingle: all alone  [2023-11-25 23:02:57,292: INFO/MainProcess] celery@DESKTOP-ULH9S6I ready. |
| Web interface for celeryEnvironment Pip install flower  $ Trading\_app> celery -A tasks.tasks:celery flower  (venv) PS D:\GB\pythonProject\Trading\_app> celery -A tasks.tasks:celery flower [I 231125 23:32:40 command:168] Visit me at http://0.0.0.0:5555 [I 231125 23:32:40 command:176] Broker: redis://localhost:6379// [I 231125 23:32:40 command:177] Registered tasks:   ['celery.accumulate',   'celery.backend\_cleanup',   'celery.chain',   'celery.chord',   'celery.chord\_unlock',   'celery.chunks',   'celery.group',   'celery.map',  'celery.starmap',  'tasks.tasks.send\_email\_report\_dashboard'] [I 231125 23:32:40 mixins:228] Connected to redis://localhost:6379// |
| Send email / simple / with background / with redis-celery from fastapi import APIRouter, BackgroundTasks, Depends from auth.base\_config import current\_user from .tasks import send\_email\_report\_dashboard  router = APIRouter(prefix="/report")   # with normal programm @router.get("/dashboard1") def get\_dashboard\_report(user=Depends(current\_user)):  # 1400 ms - Клиент ждет  send\_email\_report\_dashboard(user.username) # usual subprogramm sendemail  return {  "status": 200,  "data": "Письмо отправлено",  "details": None }  # with background tasks @router.get("/dashboard2") def get\_dashboard\_report(background\_tasks: BackgroundTasks, user=Depends(current\_user)):  # 500 ms - Задача выполняется на фоне FastAPI в event loop'е или в другом треде  background\_tasks.add\_task(send\_email\_report\_dashboard, user.username) # background task in fastapi  return {  "status": 200,  "data": "Письмо отправлено",  "details": None }  # with redis > celery @router.get("/dashboard3") def get\_dashboard\_report(user=Depends(current\_user)):  # 600 ms - Задача выполняется воркером Celery в отдельном процессе  send\_email\_report\_dashboard.delay(user.username)  return {  "status": 200,  "data": "Письмо отправлено",  "details": None } |
| Celery делает слепок кода при создании, нужне restart celery при изменении кода tasks:tasks  Trading\_app> celery -A tasks.tasks:celery worker --loglevel=INFO --pool=solo |
| Lesson10 |