Course FastAPI

Table of Content

[1 REPO 3](#_Toc152330711)

[2 Lesson0 Environment: 3](#_Toc152330712)

[3 Lesson1 Application, documentation 3](#_Toc152330713)

[4 Lesson2a 5](#_Toc152330714)

[4.1 New app 5](#_Toc152330715)

[5 Lesson2 Эндпоинты, Параметры URL и Запроса 7](#_Toc152330716)

[5.2 Get Request with 2 parameters 7](#_Toc152330717)

[5.3 Post query: change name of user: 9](#_Toc152330718)

[6 Lesson3 FastAPI - Валидация данных с Pydantic #3 9](#_Toc152330719)

[6.2 New class + datamodel 10](#_Toc152330720)

[6.3 Run uvicorn main:app –reload 10](#_Toc152330721)

[6.4 Полная структура данных 14](#_Toc152330722)

[6.5 If need that Customer will see validation Error: 17](#_Toc152330723)

[7 Lesson4 Базы данных и миграции Alembic 18](#_Toc152330724)

[7.1 Install postgress for windows 18](#_Toc152330725)

[7.2 Environment 18](#_Toc152330726)

[7.3 Models 18](#_Toc152330727)

[8 Lesson #8 Caching with Redis 18](#_Toc152330728)

[8.1 Environment 18](#_Toc152330729)

[8.2 Cache in FastAPI with Redis 20](#_Toc152330730)

[8.3 Environment pip install fastapi-cache2[redis] 20](#_Toc152330731)

[8.4 Run app 20](#_Toc152330732)

[8.5 Test cache : 20](#_Toc152330733)

[9 Lesson #9 Фоновые задачи c Celery Redis Flower 22](#_Toc152330734)

[9.1 Регистрация приложения в Google: 2-факт аутентификация – username – password 22](#_Toc152330735)

[9.2 Environment : Celery, Flower 22](#_Toc152330736)

[9.3 Инициализация Celery : 22](#_Toc152330737)

[9.4 run celery in new terminal 23](#_Toc152330738)

[10 Lesson10 25](#_Toc152330739)

|  |
| --- |
| 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 from datetime import datetime from sqlalchemy import MetaData, Table, Column, Integer, String, TIMESTAMP, ForeignKey, JSON  metadata = MetaData()  roles = Table(  "roles",  metadata,  Column("id", Integer, primary\_key=True),  Column("name", String, nullable=False),  Column("permissions", JSON) )  users = Table(  "users",  metadata,  Column("id", Integer, primary\_key=True),  Column("email", String, nullable=False),  Column("username", String, nullable=False),  Column("password", String, nullable=False),  Column("registered\_at", TIMESTAMP, default=datetime.utcnow),  Column("role\_id", Integer, ForeignKey("roles.id")), ) |
| Создание таблиц в БДEnvironment For migrations: Pip install alembicСпособ1 создать таблицы – через команду: Pythone code:  engine = sqlalchemy.create\_engine(DATABASE\_URL)  metadata.create\_all(engine) Способ2 создать таблицы – через миграции: Для миграции – библиотека Alembic  > Alembic init migrations  Trading\_app> alembic init migrations  Creating directory D:\GB\pythonProject\Trading\_app\migrations ... done  Creating directory D:\GB\pythonProject\Trading\_app\migrations\versions ... done  Generating D:\GB\pythonProject\Trading\_app\alembic.ini ... done  Generating D:\GB\pythonProject\Trading\_app\migrations\env.py ... done  Generating D:\GB\pythonProject\Trading\_app\migrations\README ... done  Generating D:\GB\pythonProject\Trading\_app\migrations\script.py.mako ... done  Please edit configuration/connection/logging settings in 'D:\\GB\\pythonProject\\Trading\_app\\alembic.ini' before proceeding.   * Created dir proj/migrations * Env.py * README * Script.py.mako * Created fole alembic.ini |
| Edit alembic.ini ...  ~~sqlalchemy.url = driver://user:pass@localhost/dbname~~  sqlalchemy.url = postgresql://%(DB\_USER)s:%(DB\_PASS)s@%(DB\_HOST)s:%(DB\_PORT)s/%(DB\_NAME)s  ... |
| Edit proj/.env DB\_HOST=localhost  DB\_PORT=5432  DB\_NAME=postgres  DB\_USER=postgres  DB\_PASS=postgres |
| Environment ( for .env files)Pip install dotenv |
| Edit proj/config.py from dotenv import load\_dotenv import os  load\_dotenv()  DB\_HOST = os.environ.get("DB\_HOST") DB\_PORT = os.environ.get("DB\_PORT") DB\_NAME = os.environ.get("DB\_NAME") DB\_USER = os.environ.get("DB\_USER") DB\_PASS = os.environ.get("DB\_PASS") |
| Edit prj/migrations/env.py from logging.config import fileConfig  from sqlalchemy import engine\_from\_config from sqlalchemy import pool  from alembic import context from config import DB\_HOST, DB\_PORT, DB\_USER, DB\_NAME, DB\_PASS # New  # this is the Alembic Config object, which provides # access to the values within the .ini file in use. config = context.config  section = config.config\_ini\_section # New config.set\_section\_option(section, "DB\_HOST", DB\_HOST) # New config.set\_section\_option(section, "DB\_PORT", DB\_PORT) # New config.set\_section\_option(section, "DB\_USER", DB\_USER) # New config.set\_section\_option(section, "DB\_NAME", DB\_NAME) # New config.set\_section\_option(section, "DB\_PASS", DB\_PASS) # New  ...  # target\_metadata = None target\_metadata = metadata # new  ... |
| Команда создания версии миграции $ alembic revision –autogenerate -m “Database creation”  -autogenerate - сравнение текущего состояния и миграций  -m - комментарий  Error: sqlalchemy.exc.OperationalError: (psycopg2.OperationalError) connection to server at "localhost" (::1), port 5432 failed: ВАЖНО: пользователь "postgres" не прошёл проверку подлинности (по паролю)  Goto PGAdmin  Databases > create DB >  DB\_NAME=postgres\_fastapi  DB\_USER=prj\_user  DB\_PASS=prj\_user  Db\_created   Edit proj/.env DB\_HOST=localhost DB\_PORT=5432 DB\_NAME=postgres\_fastapi DB\_USER=prj\_user DB\_PASS=prj\_user $ alembic revision –autogenerate -m “Database creation” (venv) PS D:\GB\pythonProject\Trading\_app> alembic revision --autogenerate -m "database creation"  INFO [alembic.runtime.migration] Context impl PostgresqlImpl.  INFO [alembic.runtime.migration] Will assume transactional DDL.  INFO [alembic.autogenerate.compare] Detected added table 'roles'  INFO [alembic.autogenerate.compare] Detected added table 'users'  Generating D:\GB\pythonProject\Trading\_app\migrations\versions\23cc22935962\_database\_creation.py ... done  (venv) PS D:\GB\pythonProject\Trading\_app> |
| Появился файл proj/migrations/versions/23cc..........\_database\_creation/py """database creation  Revision ID: 23cc22935962 Revises:  Create Date: 2023-12-01 17:10:44.671401  """ from alembic import op import sqlalchemy as sa   # revision identifiers, used by Alembic. revision = '23cc22935962' down\_revision = None branch\_labels = None depends\_on = None   def upgrade() -> None:  # ### commands auto generated by Alembic - please adjust! ###  op.create\_table('roles',  sa.Column('id', sa.Integer(), nullable=False),  sa.Column('name', sa.String(), nullable=False),  sa.Column('permissions', sa.JSON(), nullable=True),  sa.PrimaryKeyConstraint('id')  )  op.create\_table('users',  sa.Column('id', sa.Integer(), nullable=False),  sa.Column('email', sa.String(), nullable=False),  sa.Column('username', sa.String(), nullable=False),  sa.Column('password', sa.String(), nullable=False),  sa.Column('registered\_at', sa.TIMESTAMP(), nullable=True),  sa.Column('role\_id', sa.Integer(), nullable=True),  sa.ForeignKeyConstraint(['role\_id'], ['roles.id'], ),  sa.PrimaryKeyConstraint('id')  )  # ### end Alembic commands ###   def downgrade() -> None:  # ### commands auto generated by Alembic - please adjust! ###  op.drop\_table('users')  op.drop\_table('roles')  # ### end Alembic commands ### |
| Миграция (venv) PS D:\GB\pythonProject\Trading\_app> alembic upgrade 23cc22935962 INFO [alembic.runtime.migration] Context impl PostgresqlImpl. INFO [alembic.runtime.migration] Will assume transactional DDL. INFO [alembic.runtime.migration] Running upgrade -> 23cc22935962, database creation (venv) PS D:\GB\pythonProject\Trading\_app>  alembic upgrade 23cc22935962  hash версии |
| FastAPI - Регистрация и Авторизация Пользователей #5 https://www.youtube.com/watch?v=nfueh3ei8HU&list=PLeLN0qH0-mCVQKZ8-W1LhxDcVlWtTALCS&index=6 |
| Библиотека fastapi-users<https://fastapi-users.github.io/fastapi-users/12.1/> <https://github.com/fastapi-users/fastapi-users>  Environment:  <https://fastapi-users.github.io/fastapi-users/12.1/installation/>  With SQLAlchemy support¶ pip install 'fastapi-users[sqlalchemy]' |
| Configuration  Выбираем стратегию Coockie + JWT ( хранение токена в браузере) <https://fastapi-users.github.io/fastapi-users/12.1/configuration/overview/> |
| JWT JSON Web Tokens <https://jwt.io/> |
| Web FastAPI-USER  > menu > User model and databases > SQLAlchemyAsynchronous driver For PostgreSQL: pip install asyncpg  For SQLite: pip install aiosqlite Examples of DB\_URLs are: PostgreSQL: engine = create\_engine('postgresql+asyncpg://user:password@host:port/name')  SQLite: engine = create\_engine('sqlite+aiosqlite:///name.db') |
| Create auth app Create dir proj/auth  Create file prj/auth/database.py  Copy code “Create User Model” from <https://fastapi-users.github.io/fastapi-users/12.1/configuration/databases/sqlalchemy/>  Edit models.py , чтобы соответствовало database.py  Models.py  user = Table(  "user",  metadata,  Column("id", Integer, primary\_key=True),  Column("email", String, nullable=False),  Column("username", String, nullable=False),  Column("hashed\_password", String, nullable=False),  Column("registered\_at", TIMESTAMP, default=datetime.utcnow),  Column("role\_id", Integer, ForeignKey(role.c.id)),  Column("is\_active", Boolean, nullable=False),  Column("is\_superuser", Boolean, nullable=False),  Column("is\_verified", Boolean, nullable=False), ) |
| Auth/database.py from datetime import datetime from typing import AsyncGenerator  from fastapi import Depends from fastapi\_users.db import SQLAlchemyBaseUserTableUUID, SQLAlchemyUserDatabase from fastapi\_users\_db\_sqlalchemy import SQLAlchemyBaseUserTable from sqlalchemy import Column, String, Boolean, Integer, ForeignKey, TIMESTAMP from sqlalchemy.ext.asyncio import AsyncSession, async\_sessionmaker, create\_async\_engine from sqlalchemy.orm import DeclarativeBase  # DATABASE\_URL = "sqlite+aiosqlite:///./test.db" this is fro sqlite from sqlalchemy.sql import roles  from config import DB\_USER, DB\_PASS, DB\_HOST, DB\_PORT, DB\_NAME  DATABASE\_URL=f'postgresql+asyncpg://{DB\_USER}:{DB\_PASS}@{DB\_HOST}:{DB\_PORT}/{DB\_NAME}'    class Base(DeclarativeBase):  pass  # if UUID useing # class User(SQLAlchemyBaseUserTableUUID, Base): # pass class User(SQLAlchemyBaseUserTable[int], Base):  # below imported from models.User  id=Column(Integer, primary\_key=True)  email=Column(String, nullable=False)  username=Column(String, nullable=False)  password=Column(String, nullable=False)  registered\_at=Column(TIMESTAMP, default=datetime.utcnow)  role\_id=Column(Integer, ForeignKey(roles.c.id))  # below imported from SQLAlchemyBaseUserTable  hashed\_password: str = Column(String(length=1024), nullable=False)  is\_active: bool = Column(Boolean, default=True, nullable=False)  is\_superuser: bool = Column(Boolean, default=False, nullable=False)  is\_verified: bool = Column(Boolean, default=False, nullable=False)    engine = create\_async\_engine(DATABASE\_URL) async\_session\_maker = async\_sessionmaker(engine, expire\_on\_commit=False)   async def create\_db\_and\_tables():  async with engine.begin() as conn:  await conn.run\_sync(Base.metadata.create\_all)   async def get\_async\_session() -> AsyncGenerator[AsyncSession, None]:  async with async\_session\_maker() as session:  yield session   async def get\_user\_db(session: AsyncSession = Depends(get\_async\_session)):  yield SQLAlchemyUserDatabase(session, User) |
| Пересоздаем таблицы в BD DB\_HOST=localhost DB\_PORT=5432 DB\_NAME=postgres\_fastapi DB\_USER=prj\_user DB\_PASS=prj\_user  Delete tables in Dbeaver  Delete migrations, delete folder migrations  $ Alembic init migrations ( created folder)  $ > alembic revision --autogenerate -m "Initial" |
|  |

|  |
| --- |
| 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 |