Код проекта IOT

## Файл: app\database.py

# app/database.py

# from sqlalchemy import create\_engine

from sqlalchemy.ext.asyncio import AsyncSession, create\_async\_engine

DATABASE\_URL = "postgresql+asyncpg://mon\_admin@127.0.0.1/monitoring"

engine = create\_async\_engine(DATABASE\_URL, echo=True)

# engine = create\_engine(DATABASE\_URL)

## Файл: app\dependencies.py

# app/dependencies

from sqlalchemy.ext.asyncio import AsyncSession, create\_async\_engine

from sqlalchemy.orm import sessionmaker

from .database import engine

def get\_db():

SessionLocal = sessionmaker(engine, class\_=AsyncSession, expire\_on\_commit=False)

db = SessionLocal()

try:

yield db

finally:

db.close()

## Файл: app\main.py

# app/main.py

from datetime import datetime

from fastapi import FastAPI, Depends, Form, HTTPException, Request

from sqlalchemy.orm import Session

from fastapi.responses import RedirectResponse

from fastapi.staticfiles import StaticFiles

from fastapi.templating import Jinja2Templates

from werkzeug.security import check\_password\_hash

from .dependencies import get\_db

from .models import (

Group, User, Equipment,

AlertsSubscription, Workflow, AnswersList

)

app = FastAPI()

templates = Jinja2Templates(directory="templates")

app.mount("/static", StaticFiles(directory="static"), name="static")

@app.get("/select-group")

def select\_group(request: Request, db: Session = Depends(get\_db)):

"""

Отображает список групп для выбора пользователем.

"""

groups = db.query(Group).all()

return templates.TemplateResponse("select\_group.html", {"request": request, "groups": groups})

@app.post("/set-group")

async def set\_group(group\_id: int, request: Request, db: Session = Depends(get\_db)):

"""

Устанавливает выбранную группу в сессии пользователя и перенаправляет на страницу входа.

"""

request.session['group\_id'] = group\_id

return RedirectResponse(url="/select-user", status\_code=303)

@app.get("/select-user")

async def select\_user(request: Request, db: Session = Depends(get\_db)):

group\_id = request.session.get('group\_id')

if not group\_id:

raise HTTPException(status\_code=400, detail="Группа не выбрана")

users = db.query(User).filter(User.group\_id == group\_id).all()

return templates.TemplateResponse("select\_user.html", {"request": request, "users": users, "group\_id": group\_id})

@app.get("/login")

async def login\_form(request: Request):

"""

Представляет форму входа, убеждаясь, что пользователь выбран.

"""

username = request.query\_params.get('username')

if not username:

raise HTTPException(status\_code=400, detail="Пользователь не выбран")

group\_id = request.session.get('group\_id')

return templates.TemplateResponse("login.html", {"request": request, "username": username, "group\_id": group\_id})

@app.post("/login")

async def login(

request: Request,

username: str = Form(...),

password: str = Form(...),

group\_id: int = Form(...),

db: Session = Depends(get\_db)

):

"""

Аутентификация пользователя по имени пользователя и паролю.

"""

user = db.query(User).filter(User.user\_name == username, User.group\_id == group\_id).first()

if not user or not check\_password\_hash(user.user\_password, password): # Используем функцию проверки хэша

raise HTTPException(status\_code=401, detail="Неверное имя пользователя или пароль")

request.session['user\_id'] = user.user\_id

return RedirectResponse(url=f"/dashboard/{group\_id}", status\_code=303)

@app.get("/dashboard/{group\_id}")

async def dashboard(group\_id: int, request: Request, db: Session = Depends(get\_db)):

"""

Отображает панель управления со всем оборудованием, связанным с выбранной группой.

"""

equipments = db.query(Equipment).filter(Equipment.group\_id == group\_id).all()

return templates.TemplateResponse("dashboard.html", {"request": request, "equipments": equipments})

def get\_current\_user(request: Request):

"""

Получает ID текущего пользователя из сессии.

"""

user\_id = request.session.get('user\_id')

if not user\_id:

raise HTTPException(status\_code=400, detail="Пользователь не вошел в систему")

return user\_id

@app.post("/toggle-equipment/{equipment\_id}")

async def toggle\_equipment(equipment\_id: int, user\_id: str = Depends(get\_current\_user), db: Session = Depends(get\_db)):

"""

Асинхронно переключает статус активности оборудования для пользователя.

"""

subscription = await db.query(AlertsSubscription).filter(AlertsSubscription.equipment\_id == equipment\_id, AlertsSubscription.user\_id == user\_id).first()

if subscription and subscription.active:

subscription.active = False

subscription.unsubscribe\_time = datetime.now()

elif subscription:

subscription.active = True

subscription.subscribe\_time = datetime.now()

else:

subscription = AlertsSubscription(

equipment\_id=equipment\_id,

user\_id=user\_id,

active=True,

subscribe\_time=datetime.now(),

minutes\_to\_live=480

)

db.add(subscription)

await db.commit()

return {"status": "success", "active": subscription.active, "equipment\_id": equipment\_id}

@app.get("/downtimes/{equipment\_id}")

async def get\_downtimes(equipment\_id: int, db: Session = Depends(get\_db)):

"""

Получает список простоев для указанного оборудования.

"""

downtimes = db.query(Workflow).filter(Workflow.equipment\_id == equipment\_id).all()

return [{

"id": downtime.id,

"equipment\_id": downtime.equipment\_id,

"start\_id": downtime.start\_id.strftime("%Y-%m-%d %H:%M:%S"),

"stop\_id": downtime.stop\_id.strftime("%Y-%m-%d %H:%M:%S") if downtime.stop\_id else None,

"answer\_id": downtime.answer\_id

} for downtime in downtimes]

@app.post("/update-downtime/{downtime\_id}")

async def update\_downtime(downtime\_id: int, answer\_id: int, db: Session = Depends(get\_db)):

"""

Обновляет простой, связывая его с ответом оператора.

"""

downtime = db.query(Workflow).filter(Workflow.id == downtime\_id).first()

if downtime:

downtime.answer\_id = answer\_id

db.commit()

return {"status": "success", "message": "Downtime updated"}

else:

raise HTTPException(status\_code=404, detail="Downtime not found")

@app.get("/answers")

async def get\_answers(db: Session = Depends(get\_db)):

"""

Возвращает список всех возможных ответов.

"""

answers = db.query(AnswersList).all()

return [{

"answer\_id": answer.answer\_id,

"answer\_text": answer.answer\_text

} for answer in answers]

## Файл: app\main\_\_old.py

# app/main.py

from datetime import datetime

from fastapi import FastAPI, Depends, HTTPException, Form, Request

from fastapi.responses import HTMLResponse

from fastapi.staticfiles import StaticFiles

from fastapi.templating import Jinja2Templates

from sqlalchemy.orm import Session

from .dependencies import get\_db

from .database import engine

from .models import Base, UserEquipment

app = FastAPI()

# Создание таблиц

Base.metadata.create\_all(engine)

app.mount("/static", StaticFiles(directory="static"), name="static")

templates = Jinja2Templates(directory="templates")

@app.post("/login")

async def login(

username: str = Form(...),

password: str = Form(...),

db: Session = Depends(get\_db)

):

query = "SELECT \* FROM users WHERE username = :username AND password = :password" # noqa

result = db.execute(

query, {'username': username, 'password': password}

).fetchone()

if result:

return {"user\_id": result.user\_id}

else:

raise HTTPException(status\_code=400, detail="Invalid credentials")

@app.get("/equipment/")

def list\_equipment(db: Session = Depends(get\_db)):

result = db.execute("SELECT \* FROM equipment")

return [dict(row) for row in result]

@app.get("/login/", response\_class=HTMLResponse)

def get\_login(request: Request):

return templates.TemplateResponse("login.html", {"request": request})

@app.get("/dashboard/", response\_class=HTMLResponse)

def get\_dashboard(request: Request):

return templates.TemplateResponse("dashboard.html", {"request": request})

@app.post("/start\_shift/")

async def start\_shift(user\_id: int, db: Session = Depends(get\_db)):

current\_time = datetime.now()

db.execute(

"INSERT INTO user\_sessions (user\_id, check\_in\_time) VALUES (:user\_id, :time)", # noqa

{'user\_id': user\_id, 'time': current\_time}

)

db.commit()

return {"message": "Shift started", "time": current\_time}

@app.post("/end\_shift/")

async def end\_shift(user\_id: int, db: Session = Depends(get\_db)):

current\_time = datetime.now()

db.execute(

"UPDATE user\_sessions SET check\_out\_time = :time WHERE user\_id = :user\_id", # noqa

{'time': current\_time, 'user\_id': user\_id}

)

db.commit()

return {"message": "Shift ended", "time": current\_time}

@app.post("/assign\_downtime/")

async def assign\_downtime(

user\_id: int, downtime\_id: int, type\_id: int, db: Session = Depends(get\_db)

):

db.execute(

"UPDATE workflow SET answer\_id = :type\_id WHERE id = :downtime\_id AND equipment\_id IN (SELECT equipment\_id FROM user\_equipment WHERE user\_id = :user\_id)", # noqa

{'type\_id': type\_id, 'downtime\_id': downtime\_id, 'user\_id': user\_id}

)

db.commit()

return {

"message": "Downtime assigned",

"downtime\_id": downtime\_id,

"type\_id": type\_id

}

@app.post("/assign\_equipment/")

async def assign\_equipment(

user\_id: int, equipment\_id: int, db: Session = Depends(get\_db)

):

new\_assignment = UserEquipment(

user\_id=user\_id, equipment\_id=equipment\_id, start\_time=datetime.now()

)

db.add(new\_assignment)

db.commit()

return {

"message": "Equipment assigned successfully",

"user\_id": user\_id,

"equipment\_id": equipment\_id

}

@app.post("/release\_equipment/")

async def release\_equipment(

user\_id: int, equipment\_id: int, db: Session = Depends(get\_db)

):

assignment = db.query(UserEquipment).filter(

UserEquipment.user\_id == user\_id,

UserEquipment.equipment\_id == equipment\_id

).first()

if assignment:

assignment.end\_time = datetime.now()

db.commit()

return {

"message": "Equipment released",

"user\_id": user\_id,

"equipment\_id": equipment\_id

}

else:

raise HTTPException(status\_code=404, detail="Assignment not found")

## Файл: app\models.py

# coding: utf-8

from sqlalchemy import BigInteger, Boolean, CHAR, Column, DateTime, Float, ForeignKey, Index, Integer, SmallInteger, String, Table, Text, Time, text

from sqlalchemy.dialects.postgresql import OID, TIMESTAMP

from sqlalchemy.orm import relationship

from sqlalchemy.ext.declarative import declarative\_base

Base = declarative\_base()

metadata = Base.metadata

class Alert(Base):

\_\_tablename\_\_ = 'alerts'

\_\_table\_args\_\_ = (

Index('unique\_equipment\_user\_start', 'equipment\_id', 'start\_id', 'user\_id', unique=True),

)

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('alerts\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

start\_id = Column(BigInteger, nullable=False)

user\_id = Column(CHAR(32), nullable=False)

open\_time = Column(DateTime, nullable=False, server\_default=text("timezone('utc'::text, now())"))

close\_time = Column(DateTime)

answer\_id = Column(Integer)

alarm\_type = Column(Integer, nullable=False, server\_default=text("0"))

minutes\_to\_live = Column(Integer, server\_default=text("30"))

class AlertsSubscription(Base):

\_\_tablename\_\_ = 'alerts\_subscription'

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('alerts\_subscription\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

user\_id = Column(CHAR(32), nullable=False)

active = Column(Boolean, nullable=False, server\_default=text("true"))

subscribe\_time = Column(DateTime, nullable=False, server\_default=text("timezone('utc'::text, now())"))

unsubscribe\_time = Column(DateTime)

minutes\_to\_live = Column(Integer, nullable=False, server\_default=text("480"))

subscribe\_action = Column(Integer, server\_default=text("0"))

t\_all\_db\_volume = Table(

'all\_db\_volume', metadata,

Column('total', Text)

)

class AnswersCategory(Base):

\_\_tablename\_\_ = 'answers\_categories'

answer\_category = Column(Integer, primary\_key=True)

name = Column(Text)

t\_bad\_workflows = Table(

'bad\_workflows', metadata,

Column('dt', DateTime(True)),

Column('equipment\_id', BigInteger),

Column('bad\_start\_id', BigInteger),

Column('bad\_stop\_id', BigInteger),

Column('duration1', BigInteger),

Column('start\_id', BigInteger),

Column('stop\_id', BigInteger),

Column('duration2', BigInteger)

)

class BotUser(Base):

\_\_tablename\_\_ = 'bot\_users'

chat\_id = Column(BigInteger, primary\_key=True)

first\_name = Column(String(200))

last\_name = Column(String(200))

user\_phone = Column(String(50))

user\_email = Column(String(200))

class BotUsersGroup(Base):

\_\_tablename\_\_ = 'bot\_users\_groups'

group\_id = Column(Integer, primary\_key=True, nullable=False)

chat\_id = Column(BigInteger, primary\_key=True, nullable=False)

class Channel(Base):

\_\_tablename\_\_ = 'channels'

channel\_id = Column(Integer, primary\_key=True)

channel\_name = Column(String(50))

channel\_units = Column(String(50))

channel\_description = Column(String(250))

class Equipment(Base):

\_\_tablename\_\_ = 'equipment'

equipment\_id = Column(Integer, primary\_key=True)

group\_id = Column(Integer)

equipment\_name = Column(String(200))

equipment\_status = Column(Integer, nullable=False, server\_default=text("0"))

plan\_val = Column(Float(53))

mac\_address = Column(String(50))

use\_align\_filter = Column(Boolean, server\_default=text("false"))

align\_filter\_secs = Column(BigInteger, server\_default=text("15"))

std\_window\_secs = Column(BigInteger, server\_default=text("5"))

sort\_order = Column(Integer, server\_default=text("0"))

t\_equipment\_and\_groups = Table(

'equipment\_and\_groups', metadata,

Column('equipment\_id', Integer),

Column('equipment\_name', String(200)),

Column('group\_id', Integer),

Column('group\_name', String(200)),

Column('channel\_id', Integer),

Column('channel\_alias', String),

Column('is\_active', Boolean),

Column('sens\_level', Float(53)),

Column('use\_std', Boolean),

Column('std\_level', Float),

Column('mac\_address', String(50))

)

class EquipmentChannelsTest(Base):

\_\_tablename\_\_ = 'equipment\_channels\_test'

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('equipment\_channels\_test\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False)

sens\_level = Column(Float(53))

is\_active = Column(Boolean, nullable=False, server\_default=text("true"))

mac\_address = Column(String(50))

class Group(Base):

\_\_tablename\_\_ = 'groups'

group\_id = Column(Integer, primary\_key=True)

parent\_id = Column(Integer)

group\_name = Column(String(200))

group\_status = Column(Integer, nullable=False, server\_default=text("1"))

class Message(Base):

\_\_tablename\_\_ = 'messages'

message\_id = Column(CHAR(32), primary\_key=True)

chat\_id = Column(CHAR(32))

send\_time = Column(DateTime, nullable=False, server\_default=text("timezone('utc'::text, now())"))

from\_id = Column(CHAR(32), nullable=False)

to\_id = Column(CHAR(32), nullable=False)

message\_text = Column(Text)

message\_action = Column(SmallInteger)

message\_status = Column(SmallInteger, nullable=False)

class MinutedDatum(Base):

\_\_tablename\_\_ = 'minuted\_data'

moment = Column(DateTime, primary\_key=True, nullable=False)

equipment\_id = Column(Integer, primary\_key=True, nullable=False)

channel\_id = Column(Integer, primary\_key=True, nullable=False, server\_default=text("1"))

avg\_val = Column(Float(53))

values\_count = Column(Integer)

values\_array = Column(Text)

class MinutedDataTest(Base):

\_\_tablename\_\_ = 'minuted\_data\_test'

moment = Column(DateTime, primary\_key=True, nullable=False)

equipment\_id = Column(Integer, primary\_key=True, nullable=False)

channel\_id = Column(Integer, primary\_key=True, nullable=False, server\_default=text("1"))

avg\_val = Column(Float(53))

values\_count = Column(Integer)

values\_array = Column(Text)

high\_count = Column(Integer)

high\_level = Column(Float)

class MinutedOee(Base):

\_\_tablename\_\_ = 'minuted\_oee'

moment = Column(DateTime, primary\_key=True, nullable=False)

equipment\_id = Column(Integer, primary\_key=True, nullable=False)

high\_seconds = Column(Integer)

error\_seconds = Column(Integer)

t\_minuted\_top1000 = Table(

'minuted\_top1000', metadata,

Column('moment', DateTime),

Column('group\_name', String(200)),

Column('equipment\_id', Integer),

Column('equipment\_name', String(200)),

Column('channel\_id', Integer),

Column('mac\_address', String(50)),

Column('avg\_val', Float(53)),

Column('values\_count', Integer),

Column('values\_array', Text)

)

class MonDatum(Base):

\_\_tablename\_\_ = 'mon\_data'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240616(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240616'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240617(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240617'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240618(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240618'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240619(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240619'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240620(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240620'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240621(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240621'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240622(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240622'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240623(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240623'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240624(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240624'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240625(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240625'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240626(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240626'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240627(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240627'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240628(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240628'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240629(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240629'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240630(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240630'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240701(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240701'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240702(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240702'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240703(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240703'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240704(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240704'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240705(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240705'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240706(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240706'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240707(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240707'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240708(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240708'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240709(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240709'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240710(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240710'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240711(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240711'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240712(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240712'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240713(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240713'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240714(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240714'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240715(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240715'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240716(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240716'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataP20240717(Base):

\_\_tablename\_\_ = 'mon\_data\_p\_20240717'

id = Column(BigInteger, primary\_key=True, nullable=False, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, primary\_key=True, nullable=False, index=True, server\_default=text("now()"))

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

t\_mon\_data\_part = Table(

'mon\_data\_part', metadata,

Column('table\_catalog', String),

Column('table\_schema', String),

Column('table\_name', String),

Column('table\_type', String),

Column('self\_referencing\_column\_name', String),

Column('reference\_generation', String),

Column('user\_defined\_type\_catalog', String),

Column('user\_defined\_type\_schema', String),

Column('user\_defined\_type\_name', String),

Column('is\_insertable\_into', String),

Column('is\_typed', String),

Column('commit\_action', String)

)

class MonDataTemp(Base):

\_\_tablename\_\_ = 'mon\_data\_temp'

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('mon\_data\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, nullable=False)

avg = Column(Float(53))

std = Column(Float(53))

a = Column(Float(53))

b = Column(Float(53))

memo = Column(Text)

class MonDataTest(Base):

\_\_tablename\_\_ = 'mon\_data\_test'

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('mon\_data\_test\_id\_seq'::regclass)"))

equipment\_id = Column(Integer, nullable=False)

channel\_id = Column(Integer, nullable=False, server\_default=text("1"))

val = Column(Float(53))

moment = Column(DateTime, nullable=False)

t\_new\_oee\_vs\_old\_oee = Table(

'new\_oee\_vs\_old\_oee', metadata,

Column('moment', DateTime),

Column('group\_name', String(200)),

Column('equipment\_name', String(200)),

Column('equipment\_id', Integer),

Column('new\_oee', Float(53)),

Column('old\_oee', Float(53))

)

class OeeDatum(Base):

\_\_tablename\_\_ = 'oee\_data'

moment = Column(DateTime, primary\_key=True, nullable=False)

equipment\_id = Column(Integer, primary\_key=True, nullable=False)

oee\_val = Column(Float(53))

t\_oee\_data2 = Table(

'oee\_data2', metadata,

Column('moment', DateTime),

Column('equipment\_id', Integer),

Column('oee\_val', Float(53))

)

t\_oee\_data\_backup = Table(

'oee\_data\_backup', metadata,

Column('moment', DateTime),

Column('equipment\_id', Integer),

Column('oee\_val', Float(53))

)

class OeeDataTest(Base):

\_\_tablename\_\_ = 'oee\_data\_test'

moment = Column(DateTime, primary\_key=True, nullable=False)

equipment\_id = Column(Integer, primary\_key=True, nullable=False)

oee\_val = Column(Float(53))

class ReportEmail(Base):

\_\_tablename\_\_ = 'report\_emails'

report\_email = Column(String(400), primary\_key=True)

report\_person = Column(String(400))

is\_active = Column(Boolean, nullable=False, server\_default=text("true"))

class Router(Base):

\_\_tablename\_\_ = 'routers'

router\_port = Column(Integer, primary\_key=True)

router\_name = Column(String(200), nullable=False)

router\_ip = Column(CHAR(15))

t\_tables\_volume = Table(

'tables\_volume', metadata,

Column('oid', OID),

Column('table\_schema', String),

Column('table\_name', String),

Column('row\_estimate', Float),

Column('total\_bytes', BigInteger),

Column('index\_bytes', BigInteger),

Column('toast\_bytes', BigInteger),

Column('table\_bytes', BigInteger),

Column('total', Text),

Column('index', Text),

Column('toast', Text),

Column('table', Text)

)

class Test(Base):

\_\_tablename\_\_ = 'tests'

id = Column(Integer, primary\_key=True, server\_default=text("nextval('tests\_id\_seq'::regclass)"))

moment = Column(DateTime(True))

data\_value = Column(Float)

class Tests2(Base):

\_\_tablename\_\_ = 'tests2'

id = Column(Integer, primary\_key=True, server\_default=text("nextval('tests2\_id\_seq'::regclass)"))

moment = Column(DateTime, server\_default=text("timezone('utc'::text, now())"))

data\_value = Column(Float)

class User(Base):

\_\_tablename\_\_ = 'users'

user\_id = Column(CHAR(32), primary\_key=True)

user\_name = Column(String(50))

user\_full\_name = Column(String(250))

user\_mail = Column(String(250))

user\_role = Column(Integer, nullable=False, server\_default=text("0"))

user\_auth\_type = Column(Integer, nullable=False, server\_default=text("0"))

user\_status = Column(Integer, nullable=False, server\_default=text("1"))

user\_password = Column(String(400))

salt = Column(String(400))

last\_device\_id = Column(String(400))

create\_time = Column(DateTime, server\_default=text("timezone('utc'::text, now())"))

update\_time = Column(DateTime, server\_default=text("timezone('utc'::text, now())"))

bad\_tries = Column(SmallInteger, nullable=False, server\_default=text("0"))

class UsersBk(Base):

\_\_tablename\_\_ = 'users\_bk'

user\_id = Column(CHAR(32), primary\_key=True)

user\_name = Column(String(50))

user\_full\_name = Column(String(250))

user\_mail = Column(String(250))

user\_role = Column(Integer, nullable=False, server\_default=text("0"))

user\_auth\_type = Column(Integer, nullable=False, server\_default=text("0"))

user\_status = Column(Integer, nullable=False, server\_default=text("1"))

user\_password = Column(String(400))

salt = Column(String(400))

last\_device\_id = Column(String(400))

create\_time = Column(DateTime, server\_default=text("timezone('utc'::text, now())"))

update\_time = Column(DateTime, server\_default=text("timezone('utc'::text, now())"))

class UsersDevice(Base):

\_\_tablename\_\_ = 'users\_devices'

id = Column(CHAR(32), primary\_key=True)

user\_id = Column(CHAR(32), nullable=False)

device\_id = Column(String(400), nullable=False, comment='may be not unique - for many users registered on the same device')

device\_alias = Column(String(50))

reg\_date = Column(TIMESTAMP(precision=3), server\_default=text("timezone('utc'::text, now())"))

last\_connect\_date = Column(TIMESTAMP(precision=3))

is\_active = Column(Boolean, nullable=False, server\_default=text("true"))

class UsersLog(Base):

\_\_tablename\_\_ = 'users\_log'

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('users\_log\_id\_seq'::regclass)"))

action\_time = Column(DateTime, nullable=False, server\_default=text("timezone('utc'::text, now())"))

action\_content = Column(Text)

user\_id = Column(CHAR(32), nullable=False)

t\_users\_permissions = Table(

'users\_permissions', metadata,

Column('user\_name', String(50)),

Column('user\_full\_name', String(250)),

Column('user\_mail', String(250)),

Column('system\_role', Integer),

Column('user\_auth\_type', Integer),

Column('user\_status', Integer),

Column('group\_name', String(200)),

Column('user\_role', Integer)

)

class WorkTask(Base):

\_\_tablename\_\_ = 'work\_tasks'

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('work\_tasks\_id\_seq'::regclass)"))

task\_name = Column(CHAR(400))

equipment\_id = Column(BigInteger, nullable=False)

quantities = Column(BigInteger, nullable=False)

user\_id = Column(CHAR(32))

start\_moment = Column(Time)

stop\_moment = Column(Time)

class Workflow(Base):

\_\_tablename\_\_ = 'workflow'

equipment\_id = Column(BigInteger, primary\_key=True, nullable=False)

start\_id = Column(BigInteger, primary\_key=True, nullable=False)

stop\_id = Column(BigInteger)

answer\_id = Column(Integer, server\_default=text("0"))

is\_alerted = Column(Boolean, server\_default=text("false"))

class WorkflowTest(Base):

\_\_tablename\_\_ = 'workflow\_test'

equipment\_id = Column(BigInteger, primary\_key=True, nullable=False)

start\_id = Column(BigInteger, primary\_key=True, nullable=False)

stop\_id = Column(BigInteger)

answer\_id = Column(Integer, server\_default=text("0"))

is\_alerted = Column(Boolean, server\_default=text("false"))

class AnswersList(Base):

\_\_tablename\_\_ = 'answers\_list'

answer\_id = Column(Integer, primary\_key=True)

answer\_text = Column(String(400), nullable=False)

answer\_action = Column(SmallInteger)

is\_system = Column(Boolean, server\_default=text("false"))

answer\_category = Column(ForeignKey('answers\_categories.answer\_category'), nullable=False, index=True, server\_default=text("1"))

answer\_color = Column(Text, nullable=False, server\_default=text("'#BDF4A8'::text"))

answers\_category = relationship('AnswersCategory')

class EquipmentChannel(Base):

\_\_tablename\_\_ = 'equipment\_channels'

id = Column(BigInteger, primary\_key=True, server\_default=text("nextval('equipment\_channels\_id\_seq'::regclass)"))

equipment\_id = Column(ForeignKey('equipment.equipment\_id', ondelete='CASCADE', onupdate='CASCADE', match='FULL'), nullable=False)

channel\_id = Column(ForeignKey('channels.channel\_id', ondelete='CASCADE', onupdate='CASCADE', match='FULL'), nullable=False)

sens\_level = Column(Float(53))

is\_active = Column(Boolean, nullable=False, server\_default=text("true"))

mac\_address = Column(String(50))

channel\_alias = Column(String)

use\_std = Column(Boolean, server\_default=text("false"))

std\_level = Column(Float, server\_default=text("0"))

szero = Column(Float(53), server\_default=text("0.0"))

koef = Column(Float(53), server\_default=text("1.0"))

channel = relationship('Channel')

equipment = relationship('Equipment')

class ReportEquipment(Base):

\_\_tablename\_\_ = 'report\_equipment'

equipment\_id = Column(ForeignKey('equipment.equipment\_id', ondelete='CASCADE', onupdate='CASCADE'), primary\_key=True, nullable=False)

report\_email = Column(ForeignKey('report\_emails.report\_email', ondelete='CASCADE', onupdate='CASCADE'), primary\_key=True, nullable=False)

days\_count = Column(SmallInteger, server\_default=text("7"))

rep\_hours = Column(String(250), server\_default=text("'8,16'::character varying"))

equipment = relationship('Equipment')

report\_email1 = relationship('ReportEmail')

class ReportGroup(Base):

\_\_tablename\_\_ = 'report\_group'

group\_id = Column(ForeignKey('groups.group\_id', ondelete='CASCADE', onupdate='CASCADE'), primary\_key=True, nullable=False)

report\_email = Column(ForeignKey('report\_emails.report\_email', ondelete='CASCADE', onupdate='CASCADE'), primary\_key=True, nullable=False)

hours\_count = Column(SmallInteger, server\_default=text("24"))

rep\_hours = Column(String(250), server\_default=text("'8,16'::character varying"))

group = relationship('Group')

report\_email1 = relationship('ReportEmail')

class RouterAdmin(Base):

\_\_tablename\_\_ = 'router\_admins'

router\_port = Column(ForeignKey('routers.router\_port', ondelete='CASCADE', onupdate='CASCADE'), primary\_key=True, nullable=False, index=True)

admin\_email = Column(String(400), primary\_key=True, nullable=False)

admin\_name = Column(String(400))

router = relationship('Router')

class UsersGroup(Base):

\_\_tablename\_\_ = 'users\_groups'

user\_id = Column(ForeignKey('users.user\_id', ondelete='CASCADE', onupdate='CASCADE'), primary\_key=True, nullable=False)

group\_id = Column(ForeignKey('groups.group\_id', ondelete='CASCADE', onupdate='CASCADE'), primary\_key=True, nullable=False)

user\_role = Column(Integer, nullable=False, server\_default=text("0"))

group = relationship('Group')

user = relationship('User')

## Файл: app\\_\_init\_\_.py

# app/\_\_init\_\_.py

## Файл: templates\dashboard.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Панель управления</title>

<link rel="stylesheet" href="/static/css/style.css">

</head>

<body>

<div class="dashboard-container">

<h1>Панель управления</h1>

<div id="equipment-list">

<!-- Список оборудования заполняется динамически через JavaScript -->

</div>

<script>

document.addEventListener("DOMContentLoaded", function() {

const groupId = sessionStorage.getItem('group\_id');

fetch(`/equipment/${groupId}`)

.then(response => response.json())

.then(data => {

const list = document.getElementById('equipment-list');

data.forEach(equipment => {

const item = document.createElement('div');

item.className = 'equipment-item ' + (equipment.active ? 'active' : 'inactive');

item.innerHTML = `

<span>${equipment.name}</span>

<button onclick="loadDowntimes(${equipment.id})">Show Downtimes</button>

<div id="downtimes-${equipment.id}" class="downtime-container"></div>

`;

list.appendChild(item);

});

})

.catch(error => console.error('Error loading equipment:', error));

});

function loadDowntimes(equipmentId) {

fetch(`/downtimes/${equipmentId}`)

.then(response => response.json())

.then(data => {

const container = document.getElementById(`downtimes-${equipmentId}`);

container.innerHTML = ''; // Clear previous entries

data.forEach(downtime => {

const entry = document.createElement('div');

entry.className = 'downtime-entry';

entry.id = `downtime-${downtime.id}`;

entry.innerHTML = `

Start: ${downtime.start\_id}, Stop: ${downtime.stop\_id || 'Ongoing'}

<button onclick="showAnswers(${downtime.id})">Update Reason</button>

`;

container.appendChild(entry);

});

})

.catch(error => console.error('Error loading downtimes:', error));

}

function showAnswers(downtimeId) {

fetch(`/answers`)

.then(response => response.json())

.then(data => {

const selector = document.createElement('select');

data.forEach(answer => {

const option = document.createElement('option');

option.value = answer.answer\_id;

option.text = answer.answer\_text;

selector.appendChild(option);

});

selector.onchange = function() { updateDowntime(downtimeId, this.value); }

document.querySelector(`#downtime-${downtimeId}`).appendChild(selector);

})

.catch(error => console.error('Error loading answers:', error));

}

function updateDowntime(downtimeId, answerId) {

fetch(`/update-downtime/${downtimeId}`, {

method: 'POST',

headers: {'Content-Type': 'application/json'},

body: JSON.stringify({answer\_id: answerId})

})

.then(response => response.json())

.then(data => {

if (data.status === 'success') {

alert('Downtime updated successfully!');

} else {

throw new Error(data.message || 'Failed to update downtime');

}

})

.catch(error => console.error('Error updating downtime:', error));

}

</script>

</div>

</body>

</html>

## Файл: templates\login.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Вход в систему</title>

<link rel="stylesheet" href="/static/css/style.css">

</head>

<body>

<div class="login-container">

<h2>Вход в систему</h2>

<form action="/login" method="post">

<input type="hidden" name="group\_id" value="{{group\_id}}" />

<div class="form-group">

<label for="username">Имя пользователя:</label>

<input type="text" id="username" name="username" required>

</div>

<div class="form-group">

<label for="password">Пароль:</label>

<input type="password" id="password" name="password" required>

</div>

<div class="form-group">

<button type="submit">Вход</button>

</div>

</form>

</div>

</body>

</html>

## Файл: templates\select\_group.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Выбор группы</title>

<link rel="stylesheet" href="/static/css/style.css"> <!-- Убедитесь, что путь к CSS правильный -->

</head>

<body>

<div class="select-group-container">

<h1>Выберите группу</h1>

<form action="/set-group" method="post">

<label for="group\_id" hidden>Выберите группу</label> <!-- Доступность: добавлен скрытый label -->

<select name="group\_id" id="group\_id" required aria-label="Выбор группы">

{% for group in groups %}

<option value="{{ group.group\_id }}">{{ group.group\_name }}</option>

{% endfor %}

</select>

<button type="submit">Продолжить</button>

</form>

</div>

</body>

</html>

## Файл: templates\select\_user.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Выбор пользователя</title>

<link rel="stylesheet" href="/static/css/style.css">

</head>

<body>

<div class="user-select-container">

<h1>Выберите пользователя</h1>

<form action="/login" method="get">

<select name="username" required>

{% for user in users %}

<option value="{{ user.user\_name }}">{{ user.user\_name }}</option>

{% endfor %}

</select>

<button type="submit">Продолжить</button>

</form>

</div>

</body>

</html>