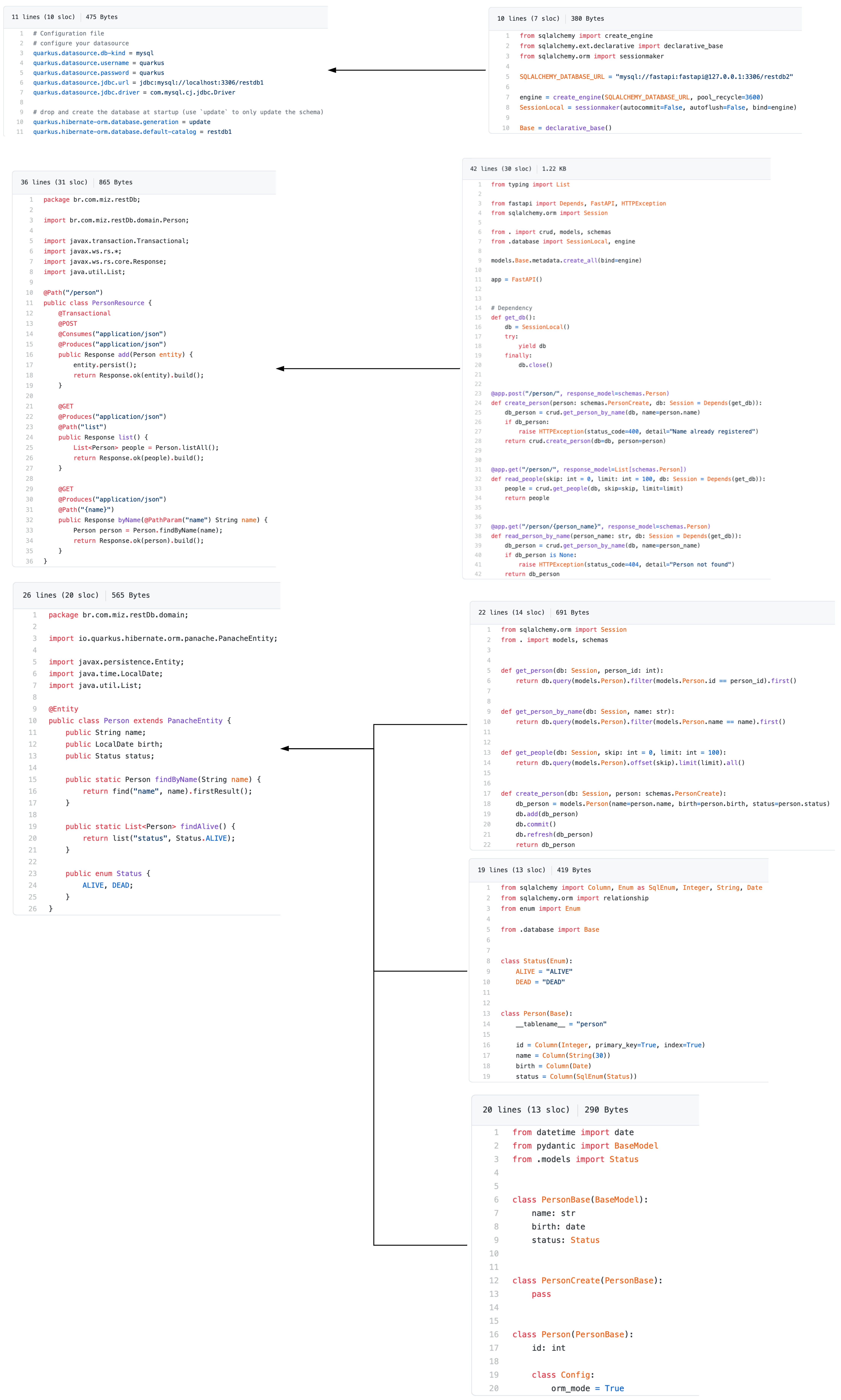
Code Complexity



11 lines (10 sloc) | 475 Bytes

```
# Configuration file
# configure your datasource
quarkus.datasource.db-kind = mysql
quarkus.datasource.username = quarkus
quarkus.datasource.password = quarkus
quarkus.datasource.jdbc.url = jdbc:mysql://localhost:3306/restdb1
quarkus.datasource.jdbc.driver = com.mysql.cj.jdbc.Driver

# drop and create the database at startup (use `update` to only update the schema)
quarkus.hibernate-orm.database.generation = update
quarkus.hibernate-orm.database.default-catalog = restdb1
```

10 lines (7 sloc) | 380 Bytes

```python
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

SQLALCHEMY_DATABASE_URL = "mysql://fastapi:fastapi@127.0.0.1:3306/restdb2"

engine = create_engine(SQLALCHEMY_DATABASE_URL, pool_recycle=3600)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()
```

36 lines (31 sloc) | 865 Bytes

```java
package br.com.miz.restDb;

import br.com.miz.restDb.domain.Person;

import javax.transaction.Transactional;
import javax.ws.rs.*;
import javax.ws.rs.core.Response;
import java.util.List;

@Path("/person")
public class PersonResource {
    @Transactional
    @POST
    @Consumes("application/json")
    @Produces("application/json")
    public Response add(Person entity) {
        entity.persist();
        return Response.ok(entity).build();
    }

    @GET
    @Produces("application/json")
    @Path("list")
    public Response list() {
        List<Person> people = Person.listAll();
        return Response.ok(people).build();
    }

    @GET
    @Produces("application/json")
    @Path("{name}")
    public Response byName(@PathParam("name") String name) {
        Person person = Person.findByName(name);
        return Response.ok(person).build();
    }
}
```

42 lines (30 sloc) | 1.22 KB

```python
from typing import List

from fastapi import Depends, FastAPI, HTTPException
from sqlalchemy.orm import Session

from . import crud, models, schemas
from .database import SessionLocal, engine

models.Base.metadata.create_all(bind=engine)

app = FastAPI()


# Dependency
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()


@app.post("/person/", response_model=schemas.Person)
def create_person(person: schemas.PersonCreate, db: Session = Depends(get_db)):
    db_person = crud.get_person_by_name(db, name=person.name)
    if db_person:
        raise HTTPException(status_code=400, detail="Name already registered")
    return crud.create_person(db=db, person=person)


@app.get("/person/", response_model=List[schemas.Person])
def read_people(skip: int = 0, limit: int = 100, db: Session = Depends(get_db)):
    people = crud.get_people(db, skip=skip, limit=limit)
    return people


@app.get("/person/{person_name}", response_model=schemas.Person)
def read_person_by_name(person_name: str, db: Session = Depends(get_db)):
    db_person = crud.get_person_by_name(db, name=person_name)
    if db_person is None:
        raise HTTPException(status_code=404, detail="Person not found")
    return db_person
```

26 lines (20 sloc) | 565 Bytes

```java
package br.com.miz.restDb.domain;

import io.quarkus.hibernate.orm.panache.PanacheEntity;

import javax.persistence.Entity;
import java.time.LocalDate;
import java.util.List;

@Entity
public class Person extends PanacheEntity {
    public String name;
    public LocalDate birth;
    public Status status;

    public static Person findByName(String name) {
        return find("name", name).firstResult();
    }

    public static List<Person> findAlive() {
        return list("status", Status.ALIVE);
    }

    public enum Status {
        ALIVE, DEAD;
    }
}
```

22 lines (14 sloc) | 691 Bytes

```python
from sqlalchemy.orm import Session
from . import models, schemas


def get_person(db: Session, person_id: int):
    return db.query(models.Person).filter(models.Person.id == person_id).first()


def get_person_by_name(db: Session, name: str):
    return db.query(models.Person).filter(models.Person.name == name).first()


def get_people(db: Session, skip: int = 0, limit: int = 100):
    return db.query(models.Person).offset(skip).limit(limit).all()


def create_person(db: Session, person: schemas.PersonCreate):
    db_person = models.Person(name=person.name, birth=person.birth, status=person.status)
    db.add(db_person)
    db.commit()
    db.refresh(db_person)
    return db_person
```

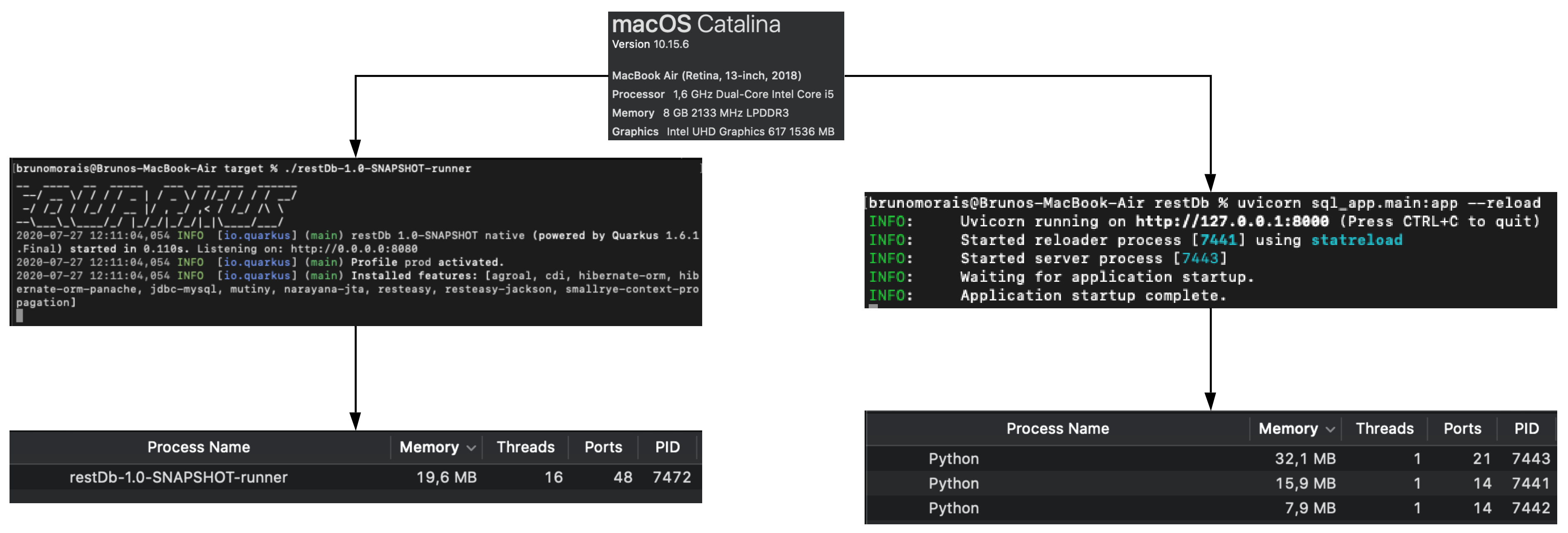
19 lines (13 sloc) | 419 Bytes

```python
from sqlalchemy import Column, Enum as SqlEnum, Integer, String, Date
from sqlalchemy.orm import relationship
from enum import Enum

from .database import Base


class Status(Enum):
    ALIVE = "ALIVE"
    DEAD = "DEAD"


class Person(Base):
    __tablename__ = "person"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(30))
    birth = Column(Date)
    status = Column(SqlEnum(Status))
```

20 lines (13 sloc) | 290 Bytes

```python
from datetime import date
from pydantic import BaseModel
from .models import Status


class PersonBase(BaseModel):
    name: str
    birth: date
    status: Status


class PersonCreate(PersonBase):
    pass


class Person(PersonBase):
    id: int

    class Config:
        orm_mode = True
```

Resources Use



macOS Catalina
Version 10.15.6

MacBook Air (Retina, 13-inch, 2018)
Processor 1,6 GHz Dual-Core Intel Core i5
Memory 8 GB 2133 MHz LPDDR3
Graphics Intel UHD Graphics 617 1536 MB

```
brunomorais@Brunos-MacBook-Air target % ./restDb-1.0-SNAPSHOT-runner
__  ____  __  _____   ___  __ ____  ______
 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
--\___\_\____/_/ |_/_/|_/_/|_|\____/___/
2020-07-27 12:11:04,054 INFO  [io.quarkus] (main) restDb 1.0-SNAPSHOT native (powered by Quarkus 1.6.1
.Final) started in 0.110s. Listening on: http://0.0.0.0:8080
2020-07-27 12:11:04,054 INFO  [io.quarkus] (main) Profile prod activated.
2020-07-27 12:11:04,054 INFO  [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, hib
ernate-orm-panache, jdbc-mysql, mutiny, narayana-jta, resteasy, resteasy-jackson, smallrye-context-pro
pagation]
```

Process Name	Memory	Threads	Ports	PID
restDb-1.0-SNAPSHOT-runner	19,6 MB	16	48	7472

```
brunomorais@Brunos-MacBook-Air restDb % uvicorn sql_app.main:app --reload
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [7441] using statreload
INFO:     Started server process [7443]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Process Name	Memory	Threads	Ports	PID
Python	32,1 MB	1	21	7443
Python	15,9 MB	1	14	7441
Python	7,9 MB	1	14	7442