■ Alembic with FastAPI and SQLModel — From Zero to Pro

Introduction

Alembic is a lightweight database migration tool used with SQLAlchemy or SQLModel. It tracks schema changes (tables, columns, indexes) and applies them safely — like version control for your database.

Project Setup Overview

We'll use FastAPI as the app framework, SQLModel (built on SQLAlchemy) as ORM, PostgreSQL as the database, and Alembic for migrations.

Step 1 — Install Dependencies

```
pip install alembic psycopg2-binary python-dotenv sqlmodel
```

Step 2 — Initialize Alembic

```
Run: alembic init alembic
This creates directories like alembic/env.py, alembic.ini, and alembic/versions.
```

Step 3 — Create .env File

```
DB_USER=postgres
DB_PASSWORD=your_password
DB_HOST=localhost
DB_PORT=5432
DB_NAME=pychemy
```

Step 4 — Configure env.py

```
from logging.config import fileConfig
from sqlalchemy import engine_from_config, pool
from alembic import context
from sqlmodel import SQLModel
from dotenv import load_dotenv
from urllib.parse import quote_plus
import os

load_dotenv()

config = context.config

DB_USER = os.getenv("DB_USER", "postgres")
DB_PASSWORD = os.getenv("DB_PASSWORD", "password")
DB_HOST = os.getenv("DB_HOST", "localhost")
DB_PORT = os.getenv("DB_PORT", "5432")
```

```
DB_NAME = os.getenv("DB_NAME", "pychemy")

encoded_password = quote_plus(DB_PASSWORD)
db_url = f"postgresql+psycopg2://{DB_USER}:{encoded_password}@{DB_HOST}:{DB_PORT}/{DB_NAME}"
config.set_main_option("sqlalchemy.url", db_url.replace("%", "%%"))

if config.config_file_name is not None:
    fileConfig(config.config_file_name)

from app.models import SQLModel
target_metadata = SQLModel.metadata

def run_migrations_offline():
    url = config.get_main_option("sqlalchemy.url")
    context.configure(url=url, target_metadata=target_metadata, literal_binds=True, compare_type
    with context.begin_transaction():
        context.run_migrations()

def run_migrations_online():
    connectable = engine_from_config(config.get_section(config.config_ini_section),
                                     prefix="sqlalchemy.", poolclass=pool.NullPool)
    with connectable.connect() as connection:
        context.configure(connection=connection, target_metadata=target_metadata, compare_type=T
        with context.begin_transaction():
            context.run_migrations()

if context.is_offline_mode():
    run_migrations_offline()
else:
    run_migrations_online()
```

Step 5 — Create Your Models

```
from sqlmodel import SQLModel, Field
from typing import Optional

class Company(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    name: str = Field(index=True)
    location: str

class Employee(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    name: str
    position: str
    company_id: int
```

Step 6 — Generate Migration Script

```
alembic revision --autogenerate -m 'create company and employee tables'
```

Step 7 — Apply Migration

```
alembic upgrade head
```

Step 8 — Update Models and Re-Migrate

```
1. Update your SQLModel class
2. Run alembic revision --autogenerate -m 'msg'
3. Run alembic upgrade head
```

Step 9 — Common Commands Reference

```
alembic init alembic
alembic revision -m "msg"
alembic revision --autogenerate -m "msg"
alembic upgrade head
alembic downgrade -1
alembic history
alembic current
alembic show <revision_id>
alembic stamp head
```

Step 10 — Troubleshooting Guide

```
Common Problems:
- Autogenerate doesn't detect models → import models in env.py
- No changes detected → ensure table=True
- Password issues → use quote_plus()
- Migration failed → alembic downgrade -1, then upgrade head
```

Summary

■ You've configured Alembic with .env and SQLModel, created and applied migrations, and learned troubleshooting methods.