■ Guía Avanzada Backend Restaurante – FastAPI + MySQL

Este documento describe la arquitectura del backend para la gestión de una carta de restaurante usando **FastAPI** y **MySQL**. Incluye el modelo de base de datos proporcionado, recomendaciones de mejora y snippets de código en Python/SQLAlchemy y Pydantic.

■■ Esquema de Base de Datos (final adaptado)

```
[Plato] --- (N:1) ---> [CategoriaPlato]
   |
   |--< (N:M) >-- [PlatoAlergeno] -- (N:1) --> [Alergeno]

[Vino] --- (N:1) ---> [CategoriaVino]
   |
   |-- (N:1) --> [Bodega]
   |-- (N:1) --> [DenominacionOrigen]
   |-- (N:1) --> [Enologo]
   |--< (N:M) >-- [VinoUva] -- (N:1) --> [Uva]
```

■ Recomendaciones aplicadas

- Uso de **DECIMAL(10,2)** para precios. - Nombres de campos en **snake_case** para consistencia. - Campos `nombre` en tablas de referencia definidos como `NOT NULL UNIQUE`. - Inclusión de registros “Desconocidos” para bodega, enólogo, DO cuando no se conozca el dato. - Índices en FKs (`categoria_id`, `plato_id`, `vino_id`) para optimizar consultas. - Semillas iniciales para categorías (`Entrantes`, `Postres`, `Tinto`, `Blanco`, etc.).

■ Modelos SQLAlchemy (ejemplo)

```
python
from sqlalchemy import Column, Integer, String, Numeric, ForeignKey, Table
from sqlalchemy.orm import relationship
from database import Base

# Tabla intermedia platos ↔ alergenos
plato_alergeno = Table(
    "plato_alergeno",
    Base.metadata,
    Column("plato_id", Integer, ForeignKey("plato.id"), primary_key=True),
    Column("alergeno_id", Integer, ForeignKey("alergeno.id"), primary_key=True)
)

# Tabla intermedia vinos ↔ uvas
vino_uva = Table(
    "vino_uva",
    Base.metadata,
    Column("vino_id", Integer, ForeignKey("vino.id"), primary_key=True),
    Column("uva_id", Integer, ForeignKey("uva.id"), primary_key=True)
)

class CategoriaPlato(Base):
    __tablename__ = "categoria_plato"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(100), unique=True, nullable=False)

class Plato(Base):
```

```
    __tablename__ = "plato"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(255), nullable=False)
    precio = Column(Numeric(10,2), nullable=False)
    descripcion = Column(String(500))
    categoria_id = Column(Integer, ForeignKey("categoria_plato.id"))
    categoria = relationship("CategoriaPlato")
    alergenos = relationship("Alergeno", secondary=plato_alergeno, back_populates="platos")

class Alergeno(Base):
    __tablename__ = "alergeno"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(100), unique=True, nullable=False)
    platos = relationship("Plato", secondary=plato_alergeno, back_populates="alergenos")

class CategoriaVino(Base):
    __tablename__ = "categoria_vino"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(100), unique=True, nullable=False)

class Bodega(Base):
    __tablename__ = "bodega"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(255), unique=True, nullable=False)

class DenominacionOrigen(Base):
    __tablename__ = "denominacion_origen"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(255), unique=True, nullable=False)

class Enologo(Base):
    __tablename__ = "enologo"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(255), unique=True, nullable=False)

class Uva(Base):
    __tablename__ = "uva"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(100), unique=True, nullable=False)
    vinos = relationship("Vino", secondary=vino_uva, back_populates="uvas")

class Vino(Base):
    __tablename__ = "vino"
    id = Column(Integer, primary_key=True, index=True)
    nombre = Column(String(255), nullable=False)
    precio = Column(Numeric(10,2), nullable=False)
    categoria_id = Column(Integer, ForeignKey("categoria_vino.id"))
    bodega_id = Column(Integer, ForeignKey("bodega.id"), nullable=True)
    denominacion_origen_id = Column(Integer, ForeignKey("denominacion_origen.id"), nullable=True)
    enologo_id = Column(Integer, ForeignKey("enologo.id"), nullable=True)

    categoria = relationship("CategoriaVino")
    bodega = relationship("Bodega")
    denominacion_origen = relationship("DenominacionOrigen")
    enologo = relationship("Enologo")
    uvas = relationship("Uva", secondary=vino_uva, back_populates="vinos")
```

■ Schemas Pydantic (ejemplo)

```
python
from pydantic import BaseModel, Field
from typing import List, Optional

class PlatoBase(BaseModel):
    nombre: str = Field(..., min_length=1, max_length=255)
    precio: float = Field(..., gt=0)
    descripcion: Optional[str] = None
```

```
    categoria_id: int

class PlatoCreate(PlatoBase):
    alergenos_ids: Optional[List[int]] = []

class PlatoRead(PlatoBase):
    id: int
    alergenos: List[str] = []

    class Config:
        orm_mode = True

class VinoBase(BaseModel):
    nombre: str
    precio: float
    categoria_id: int
    bodega_id: Optional[int] = None
    denominacion_origen_id: Optional[int] = None
    enologo_id: Optional[int] = None

class VinoCreate(VinoBase):
    uvas_ids: Optional[List[int]] = []

class VinoRead(VinoBase):
    id: int
    uvas: List[str] = []

    class Config:
        orm_mode = True
```

■ Ejemplo de Endpoint (CRUD Plato)

```
python
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from database import get_db
from models import Plato, Alergeno
from schemas import PlatoCreate, PlatoRead

router = APIRouter()

@router.post("/platos", response_model=PlatoRead)
def create_plato(plato: PlatoCreate, db: Session = Depends(get_db)):
    db_plato = Plato(
        nombre=plato.nombre,
        precio=plato.precio,
        descripcion=plato.descripcion,
        categoria_id=plato.categoria_id
    )
    if plato.alergenos_ids:
        alergenos = db.query(Alergeno).filter(Alergeno.id.in_(plato.alergenos_ids)).all()
        db_plato.alergenos = alergenos
    db.add(db_plato)
    db.commit()
    db.refresh(db_plato)
    return db_plato
```