Chiến lược & Lập luận thiết kế (giải thích vì sao chọn như trên)

A. Tại sao tách LichLamViecTemplate và Slot?

- **LichLamViecTemplate** lưu quy tắc (mẫu tuần lặp) — tiện khi bác sĩ có lịch cố định tuần này sang tuần khác (VD: thứ 2, thứ 4 09:00-12:00).
- **Slot** là instances thực tế cho từng ngày (ví dụ 2025-11-15 09:00-09:30). Việc generate slot giúp:
    - Dễ truy vấn slot trống nhanh (SELECT trên table Slot với index `MaBS, SlotDate`).
    - Cho phép override/exception (nghỉ, dời giờ) dễ dàng bằng `LichLamViecException`.
    - Hệ thống có thể pre-generate 30-90 ngày để người dùng đặt, giảm tính toán thời gian thực.

B. Vì sao cần `Slot` để quản lý đặt lịch?

- Nếu không có thực thể Slot, cần kiểm tra khoảng thời gian trống bằng logic phức tạp và dễ bị race condition. Với Slot:
    - Có ràng buộc duy nhất (unique) và index, cho phép insert atomic `LichHen` trỏ SlotID, đơn giản và an toàn.
    - Hỗ trợ các thao tác như block slot, tạm đóng, hiển thị trạng thái.

C. Tại sao tạo `LichHen` tách rời thay vì gán trực tiếp vào `Slot`?

- `Slot` là khung giờ (có thể reuse), `LichHen` lưu thông tin người đặt, trạng thái, ghi log xử lý giao dịch. Tách giúp theo dõi lịch sử booking, tracking, refunds, payments, etc.

D. Quyết định PK INT vs UUID

- Mặc định dùng `INT AUTO_INCREMENT` cho hiệu quả, nhưng nếu cần scale phân tán (microservices, multi-region), **UUID** có lợi. Nếu chọn UUID, các index lớn hơn và cần cân nhắc về hiệu suất.

Câu hỏi chiến lược:

Q1 — Thiết kế Lịch làm việc: mình thiết kế `LichLamViecTemplate` + `LichLamViecException` + `Slot` như trên. Vì sao?

Lý do & lợi ích:

1. **Linh hoạt**: Template lưu pattern tuần lặp (ví dụ T2, T4 09:00-12:00), slotDuration cho phép tạo slot 15/30/60 phút. Exception cho phép:
    - Nghỉ 1 ngày
    - Thay đổi khung giờ cho 1 ngày cụ thể
2. **Khả năng sinh slot**: Hệ thống có thể tự động generate slot cho X ngày tới, phù hợp khi bác sĩ thay đổi lịch sẽ chỉ cần cập nhật template/exception và regen slots.
3. **Hỗ trợ nhiều kịch bản**:
    - Bác sĩ làm theo tuần (template).
    - Bác sĩ có lịch đặc biệt (exception override).
    - Bác sĩ làm nhiều ca/đa phòng khám (Slot chứa MaPK để support đa phòng khám).
4. **Độ chính xác & concurrency**: Khi slot đã tồn tại trên DB, việc đặt lịch chỉ cần insert LichHen trỏ SlotID với ràng buộc UNIQUE — dễ quản lý cạnh tranh.
5. **Mở rộng**: Nếu sau này cần thêm loại slot (telemedicine), chỉ cần thêm cột `SlotType ENUM('InPerson','Telehealth')`.

Kết luận: cấu trúc đủ linh hoạt cho việc bác sĩ làm ở các khung giờ khác nhau vào các ngày khác nhau — template + exception + generated slots cho phép biểu diễn mọi pattern.

Q2 — Hiệu suất: khi hệ thống có hàng triệu lịch hẹn, tìm slot trống sao cho nhanh?

Vấn đề: truy vấn slot trống theo bác sĩ/ phòng khám/ ngày có thể chậm nếu scan table lớn.

Giải pháp đề xuất (ít nhất 3 phương án):

1. **Chỉ mục hợp lý**
    - Tạo index composite: `INDEX idx_slot_bs_date (MaBS, SlotDate, StartTime)` — để truy vấn `WHERE MaBS = ? AND SlotDate BETWEEN ? AND ? AND NOT EXISTS booking`.
    - Trên `LichHen`, index `SlotID` và `TrangThai` để lọc booking đã hủy/đã hoàn thành.
2. **Denormalize / materialized availability table**
    - Tạo bảng **SlotAvailability** hoặc materialized view chứa thông tin tóm tắt (SlotID, IsBooked boolean, LastUpdated). Cập nhật khi booking/cancel. Truy vấn slot trống chỉ truy vấn bảng nhỏ này. Có thể làm materialized view hoặc bảng cập nhật trong transaction.
    - Ưu: query cực nhanh; Nhược: cần duy trì đồng bộ.
3. **Cache (in-memory)**
    - Dùng **Redis** để cache danh sách slot trống cho mỗi bác sĩ + ngày (`key: slots:doctor:{id}:date:{yyyy-mm-dd}`).
    - Khi booking/cancel, cập nhật cache hoặc invalidate.
    - Ưu: throughput cao; Nhược: phức tạp đồng bộ trạng thái.
4. **Partitioning**
    - Partition `Slot` và `LichHen` theo `SlotDate` (range partition by month) để giảm I/O khi query theo ngày.
    - Khi dữ liệu tăng lớn, partition giúp prune partitions.
5. **SQ L Query tối ưu & pagination**
    - Query chỉ lấy các slot trong khoảng ngày cần thiết (ví dụ 7-30 ngày tiếp theo).

- Sử dụng `LIMIT` + `ORDER BY` với index.

6. **Locking & Atomicity**
    - Khi booking, dùng transaction + `SELECT ... FOR UPDATE` trên `Slot` (hoặc `INSERT INTO LichHen` và rely on UNIQUE constraint) để tránh race.
    - Option: dùng optimistic concurrency với version number hoặc compare-and-set (Redis SETNX) để reserve slot trong vài phút.