Solution Architecture

# Overview

This document address how this system requirements will be converted into a software solution. Looking at the system requirements, it is divided to four main parts.

1. Customer
2. Stylists
3. Appointment
4. Time slots.

Customer, stylist and appointment are strait forward requirements so this document will not address them. This document will concentrate on the Time slots and its relationship with customer, stylist, and appointment.

# Time slots

Time slot can be either an available or unavailable. The available time slot is the time slot that has appointments less than the number of available stylist at that particular slot. By design the number of active stylists on a time slot is equal to number of stylists defined in the system. This can be changed later and actually is an invalid requirements. For example, stylists can take lunch break, can get sick, or even later they can work at a specific shift.

Appointment is a relationship between time slot, customer and stylist. When an appointments is created. It establish a relationship only with time slot and customer.

Available stylist for a time slot is the stylist that has no appointment at the requested time slot.

In designing this solution we need to solve racing problem for available time slots and stylists.

# Racing for time slots and stylist

## Overview

Assume we have three stylists and ten customers. Assume also that the ten customers asked for an appointment on 1/5/2019 at 9:0 at the same time. So our system will receive these requested and process them at the same time. The system should always make sure that three of these request will get the appointment while the other seven will be rejected. Though this process the system should have a valid state and behave correctly.

To create an appointment we need to verify that there is an available time slot. To do that we need to compute the appointments on that time slots with the available styles on the same time slot which is quite a work and can affect the performance of the system. Now after the computation the data, system state can change (like another appointment got added, or styles is not available) and our computation become incorrect.

As a result what we want to do is we want to create appointments correctly based on available time slot and available stylists regardless of the system traffic.

## Solution

### Selection criteria

Before we provide solutions to this problem lets us identify our selection criteria.

Performance.

Always correct result.

Scalability.

Cohesion.

Coupling.

### Solutions

#### Solution1

This solution will use data layer functionalities to provide correct results. For example system can depends on data layer transaction and locks to provide correct data. But this data layer system will become bottle nick and it will be hard to scale. When the computation of available slots becomes complicated the system performance will be effected. System is note cohesive and is coupled since Adding an appointment and computing available time slots and available stylist is done in same place.

#### Solution2

This is same as solution one except no calculation is needed. So this is similar to your account balance, it is always ready and no need to calculate it. Even though it will improve performance, but it will increase coupling between systems and as system grows the complexity will grow.

#### Solution 3

This solution is based on the idea of how to provide feedback. For example, when a customer send an appointment request, we do not approve the request immediately. So we notify the user by an event approving or rejecting his request. If this is an acceptable by customers then the solution will be valid.

After an appointment request had been received, the system, will validate the request, it if is valid it will send it to a queue where it will be processed by another system. If the request is not valid user will be notified immediately. Since we are using queuing system which will be proceeded by single thread, data will be accurate and competed correctly.

System can scale by adding more queues to it. For example, we can add a 7 queues every queue will handle requests for a specific day of the week like Saturday queue, Sunday queue, and so on.

System is decoupled since appointment creation is completely decoupled from appointment approval.