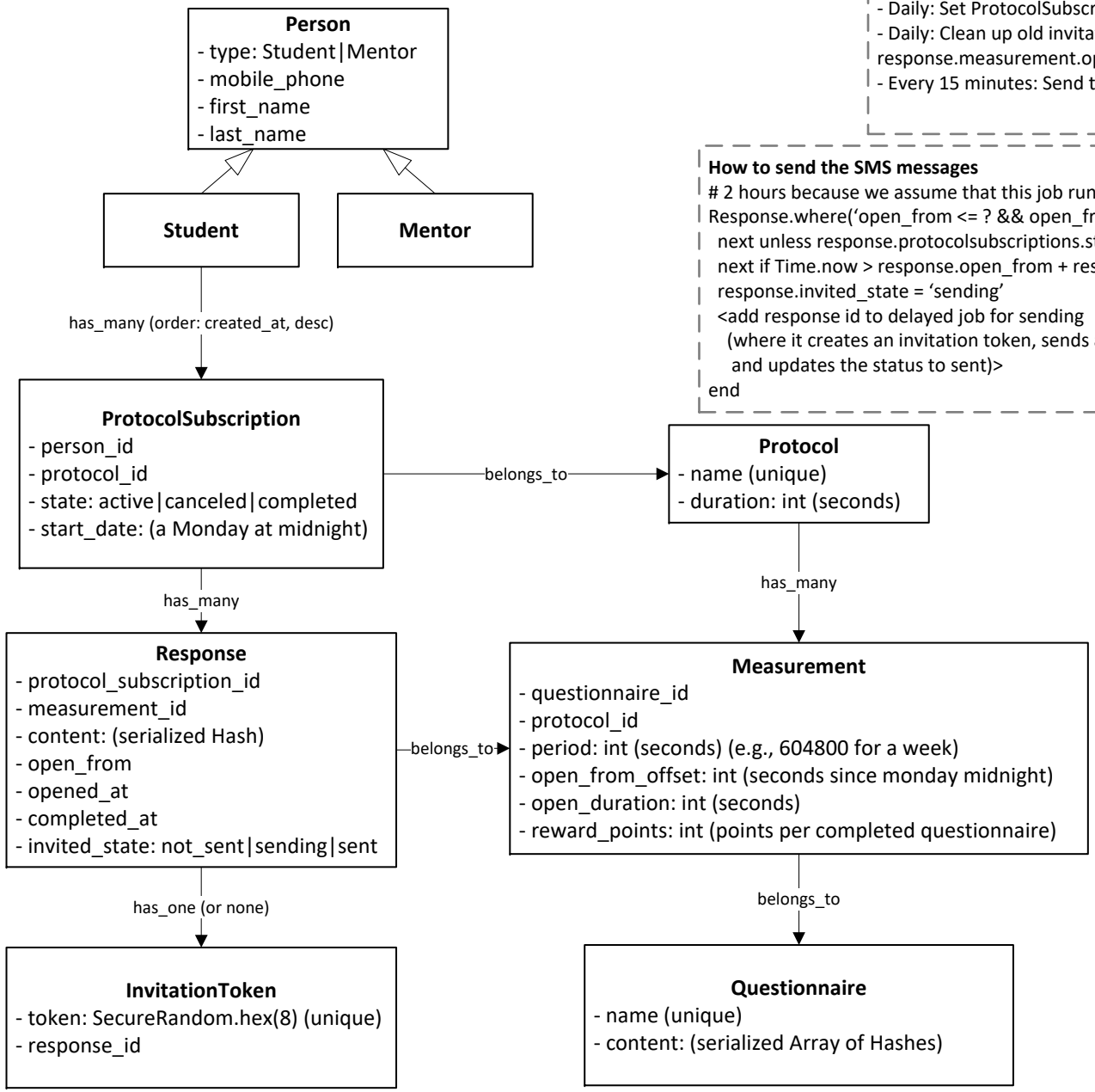
VSV Architecture



Background tasks

- Daily: Set ProtocolSubscriptions to completed if state == active && start_date + protocol.duration < Time.now.
- Daily: Clean up old invitation tokens if Time.now is > their response's open_from + response.measurement.open_duration or if that open_duration is nil, the prot_sub.start_date+protocol.duration.
- Every 15 minutes: Send the SMS messages (see below)

How to send the SMS messages

```
# 2 hours because we assume that this job runs at least once every 2 hours and otherwise we don't send invites.
Response.where('open_from <= ? && open_from > ?',Time.now, 2.hours.ago).where(invited_state: 'not_sent'). each do |response|
  next unless response.protocolsubscriptions.state == 'active'
  next if Time.now > response.open_from + response.measurement.open_duration
  response.invited_state = 'sending'
  <add response id to delayed job for sending
    (where it creates an invitation token, sends an sms to response.protocolsubscription.person.phone_number,
     and updates the status to sent)>
end
```

After creating a ProtocolSubscription, create Responses:

```
# Investigate how much daylight savings time changes screw up the schedule.
prot_sub_end = prot_sub.start_date + prot_sub.protocol.duration
ActiveRecord::Base.transaction do
  prot_sub.protocol.measurements.each do |measurement|
    open_from = prot_sub.start_date + measurement.offset
    while open_from < prot_sub_end do
      Response.create!(
        protocol_subscription_id: prot_sub.id,
        measurement_id: measurement.id,
        open_from: open_from
      )
      break unless measurement.period
      open_from += measurement.period
    end
  end
end
```

(Live) calculation of reward points for a protocol subscription:

```
max_points = 0
cur_points = 0
prot_sub.responses.each do |reponse|
  points = response.measurement.reward_points
  max_points += points
  cur_points += points if response.completed_at
end
```