
BaseEntity
{abstract}
+ Id : int
+ Equals(Object obj) : bool
+ GetHashCode() : int
- GetUnproxiedType() :Type
SchedulingAppointmentEventType
enum {STARTED, PICKED_DATE, PICKED_SPECIALITY, PICKED_DOCTOR, PICKED_TIME, FINISHED}
AppointmentState
enum {DELETED, FINISHED, PENDING, NOT_CREATED}
DomainEvent
{abstract}
+ Id : int
+ AggregateId : int
+ Timestamp : DateTime
EventSourcedAggregate
{abstract}
Version : int
+ Apply(DomainEvent changes) : void
1
*
Changes
SchedulingAppointmentDomainEvent
{abstract}
+ Type : SchedulingAppointmentEventType
Appointment
+ Id : int
+ Doctor : Doctor
+ Patient : Patient
+ Room : Room
+ StartAt : DateTime
+ EndAt : DateTime
+ State : AppointmentState
+ Apply(DomainEvent changes) : override void
SchedulingAppointmentDomainEventDTO
+ Type : SchedulingAppointmentEventType
+ AggregateId : int
+ Time : DateTime
AppointmentController[Public]
+ GetTimeIntervalsForStepByStep(string doctorUid, DateTime chosen) : Task<IActionResult>
+ GetTimeIntervalsForRecommendation(string doctorUid, DateTime start, DateTime end) : Task<IActionResult>
+ GetTimeIntervalsForRecommendationDatePriority(string speciality, DateTime start, DateTime end) : Task<IActionResult>
+ Create(CreateAppointmentForPatientDTO createAppointmentDto) : Task<IActionResult>
+ CreateInitialAppointment() : Task<IActionResult>
+ PickedDateForAppointment(DateTime startAt, int appointmentId) : Task<IActionResult>
+ PickedDoctorForAppointment(string doctorUid, int appointmentId) : Task<IActionResult>
+ PickedTimeForAppointment(TimeInterval chosenTime, int appointmentId) : Task<IActionResult>
+ ScheduleAppointment(TimeInterval chosenTime, int appointmentId) : Task<IActionResult>
- GetCurrentUser() : UserDTO
+ GetPatientsAppointments() : Task<IActionResult>
+ Cancel(int id) : Task<IActionResult>
+ GetAllFinishedPatientsAppointments() : Task<IActionResult>
+ GetPdf() : Task<IActionResult>
+ AddEvent(ScheduleAppointmentEventDTO) : Task<IActionResult>
1


1
AppointmentService
+ GetAll() : Task<IEnumerable<Appointment>>
+ Update(Appointment appointment) : void
+ GetTimeIntervalsForStepByStep(int doctorId, DateTime chosen) : Task<IEnumerable<TimeInterval>>
+ GetTimeIntervalsForRecommendationDatePriority(int doctorId, DateTime chosen) : Task<IEnumerable<TimeIntervalWithDoctorDTO>>
+ CancelAppointment(int appointmentId) : AppointmentCancelledDTO
+ CancelPatientAppointment(int appointmentId) : Appointment
+ GetUpcomingForDoctor(Doctor doctor) : Task<IEnumerable<Appointment>>
+ Schedule(Appointment appointment) : void
+ GetUpcomingAppointmentsForRoom(Room room) : Task<IEnumerable<Appointment>>
+ Create(Appointment appointment) : Task<Appointment>
+ CreateEmpty(Appointment appointment) : Task<Appointment>
+ GetAllPatientsAppointments(int patientId) : Task<IEnumerable<Appointment>>
+ Reschedule(int appointmentId,DateTime start, DateTime end) : Task<AppointmentRescheduledDTO>
+ GetAllForDoctorAndRange(int doctorId, TimeInterval interval) : Task<IEnumerable<Appointment>>
+ FormatAppointmentsForCalendar(IEnumerable<Appointment> appointments, TimeInterval interval) : IEnumerable<CalendarAppointmentsDTO>
+ GetById(int appointmentId) : Task<Appointment>
+ GetAllFinishedPatientAppointments(int patientId) : Task<IEnumerable<Appointment>>
+ GetByExamination(int examinationId) : ExaminationReport
+ GetMonthStatisticsByDoctorId(int doctorId, int month) : Task<IEnumerable<AppointmentStatisticsDTO>>
+ GetYearStatisticsByDoctorId(int doctorId) : Task<IEnumerable<AppointmentStatisticsDTO>>
+ GetTimeRangeStatisticsByDoctorId(int doctorId, TimeInterval timeinterval) : Task<IEnumerable<AppointmentStatisticsDTO>>
+ AddEvent(SchedulingAppointmentDomainEvent schedulingAppointmentDomainEvent) : void
1
1
1
TimeIntervalValidationService
+ ValidateAppointment(Appointment appointment) : Task
+ ValidateReallocation(EquipmentReallocation reallocation) : Task
+ ThrowIfEndBeforeStart(DateTime start, DateTime end) : void
+ ValidateRescheduling(Appointment appointment, DateTime start, DateTime end) : Task
+ IsIntervalOverlapingWithDoctorAppointments(int doctorId, Timeinterval possibleTimeInterval) : Task<bool>
+ IsIntervalOverlapingWithRoomsAppointments(int roomId, Timeinterval possibleTimeInterval) : Task<bool>
+ ValidateRenovation(Renovation renovation) : Task
- ThrowIfRenovationsAreOverlaping(TimeInterval timeInterval) : void
- ThrowIfNotInWorkingHours(DateTime appointmentStartAt, DateTime appointmentEndAt, int doctorId) : void
- ThrowIfNotInWorkingHours(Appointment appointment) : void
- ThrowIfInPast(DateTime start) : void
- ThrowIfIntervalsAreOverlaping(List<TimeInterval> intervals, TimeInteval ti) : void
- CompactIntervals(List<TimeInterval> intervals) : List<TimeInterval>
- JoinInervalsIfTouching(TimeInterval interval, TimeInterval dateInterval) : void
- AddFirstIfIntervalEmpty(List<TimeInterval> dateIntervals, TimeInterval Interval) : void
- AddIfGapBetweenIntervals(List<TimeInterval> Intervals, TimeInterval nterval,TimeInterval dateInterval) : void
- ThrowIfInAnnualLeavePeriod(int doctorId, timeInterval) : void

- ThrowIfInTeamBuildingPeriod(int doctorId, timeInterval) : void

1

AppointmentRepository

+ GetAllForRoom(int roomId) : Task<List<Appointment>>

+ GetAllPendingForRange(TimeInterval interval, int roomId) : Task<List<TimeInterval>>

+ GetAllRoomTakenIntervalsForDate(int roomId, DateTime date) : Task<List<TimeInterval>>

+ GetAllDoctorTakenIntervalsForDate(int doctorId, DateTime date) : Task<IEnumerable<TimeInterval>>

+ GetAllDoctorTakenIntervalsForTimeInterval(int doctorId, TimeInterval timeInterval) : Task<IEnumerable<TimeInterval>>

+ GetAllDoctorUpcomingAppointments(int doctorId) : Task<IEnumerable<Appointment>>

+ GetAllRoomUpcomingAppointments(int roomId) : Task<IEnumerable<Appointment>>

+ GetAllDoctorAppointmentsForRange(int doctorId, TimeInterval interval) : Task<IEnumerable<Appointment>>

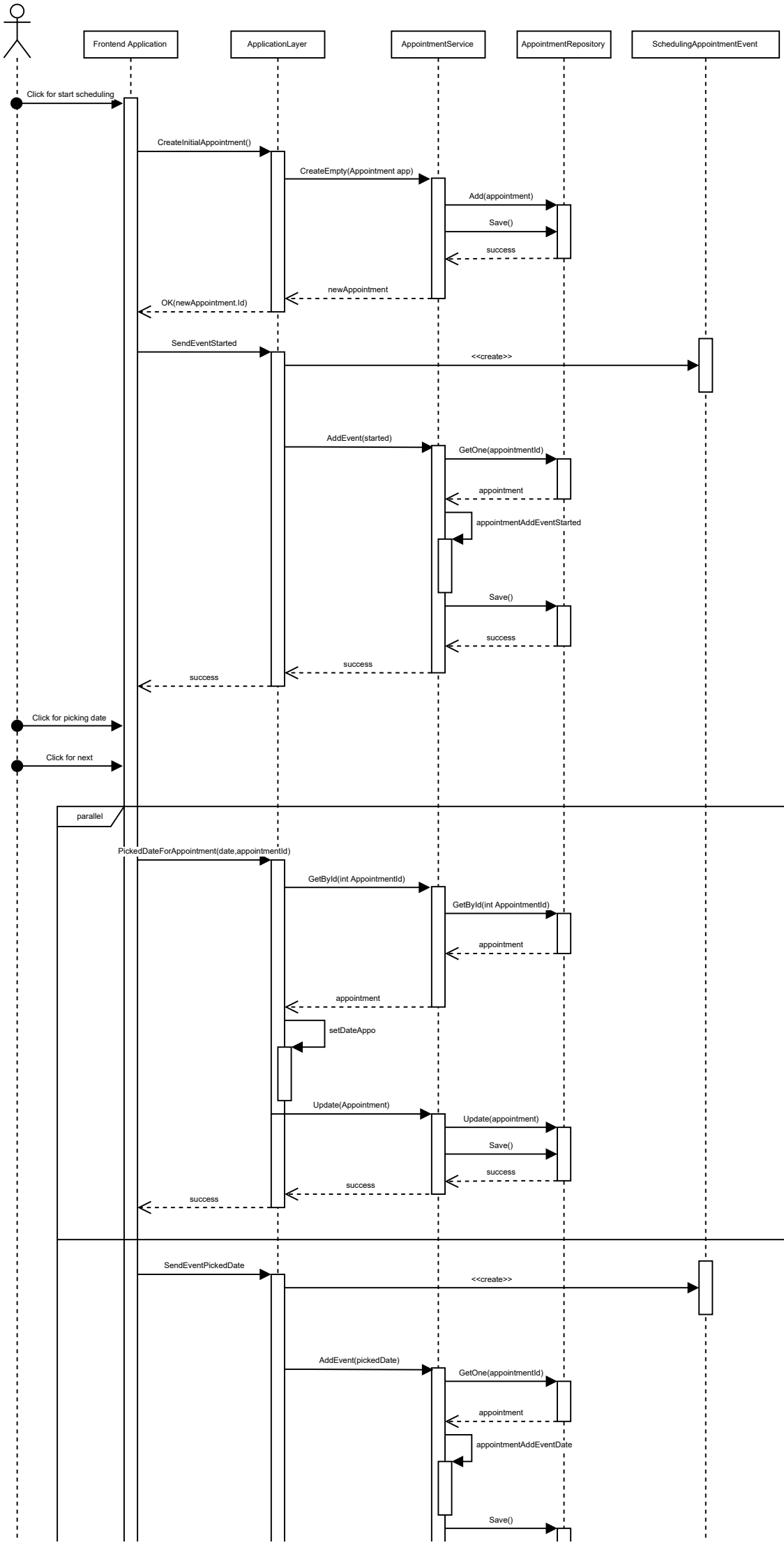
+ GetNumberOfDoctorsAppointmentsForRange(int doctorId, TimeInterval interval) : int

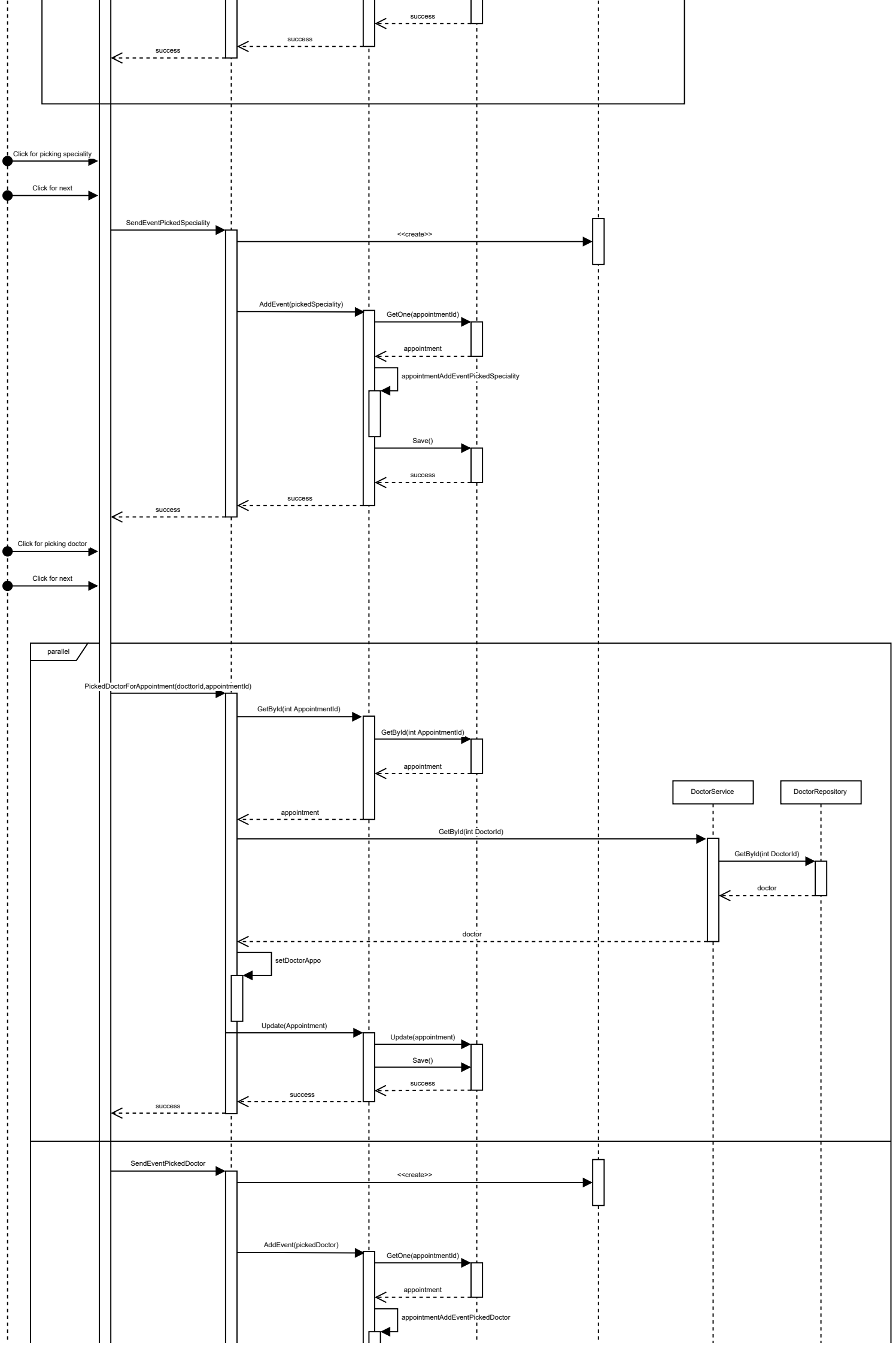
+ GetById(int appointmentId) : Task<Appointment>

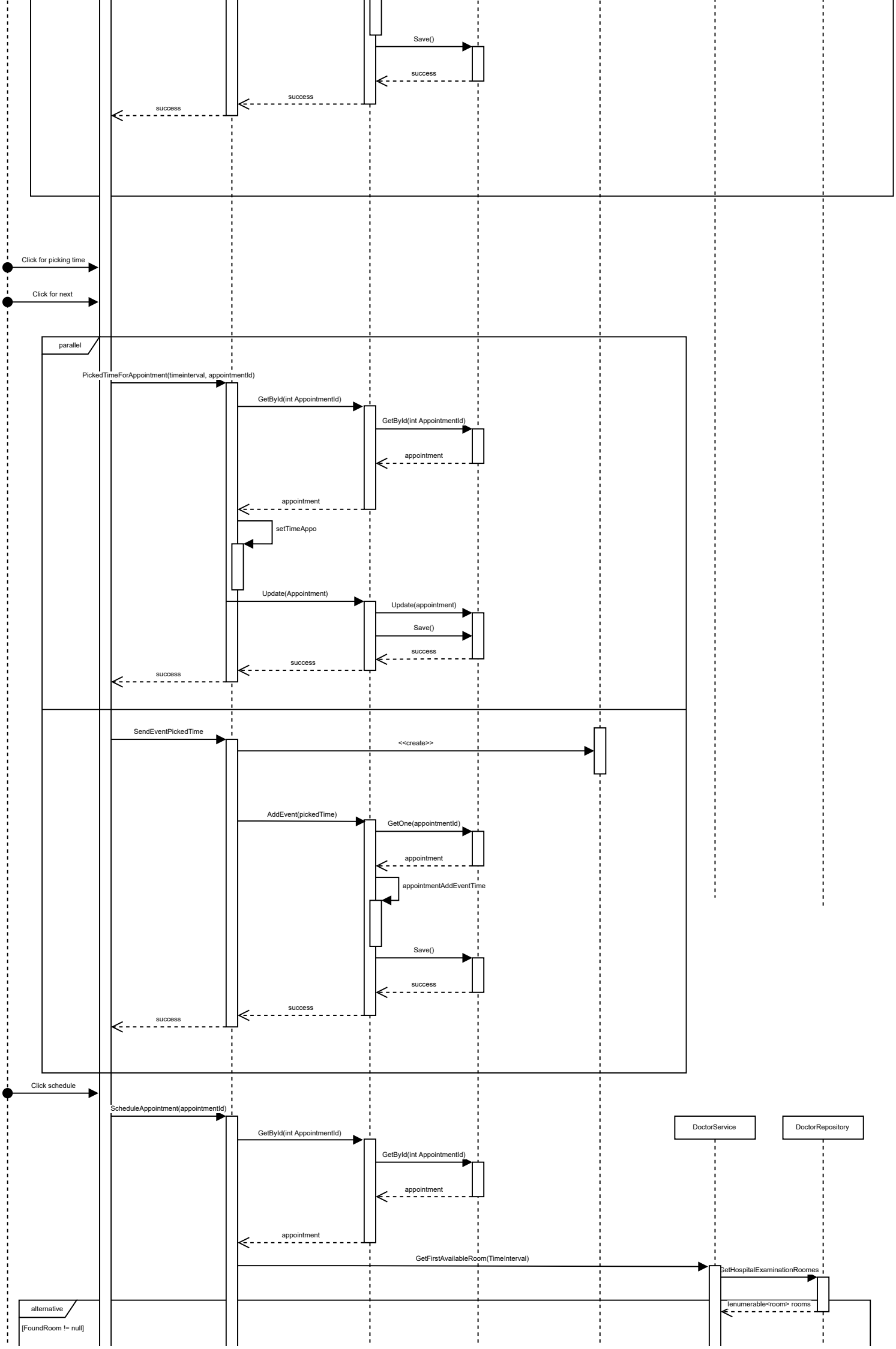
+ GetAllPatientAppointments(int patientId, TimeInterval interval) : Task<IEnumerable<Appointment>>

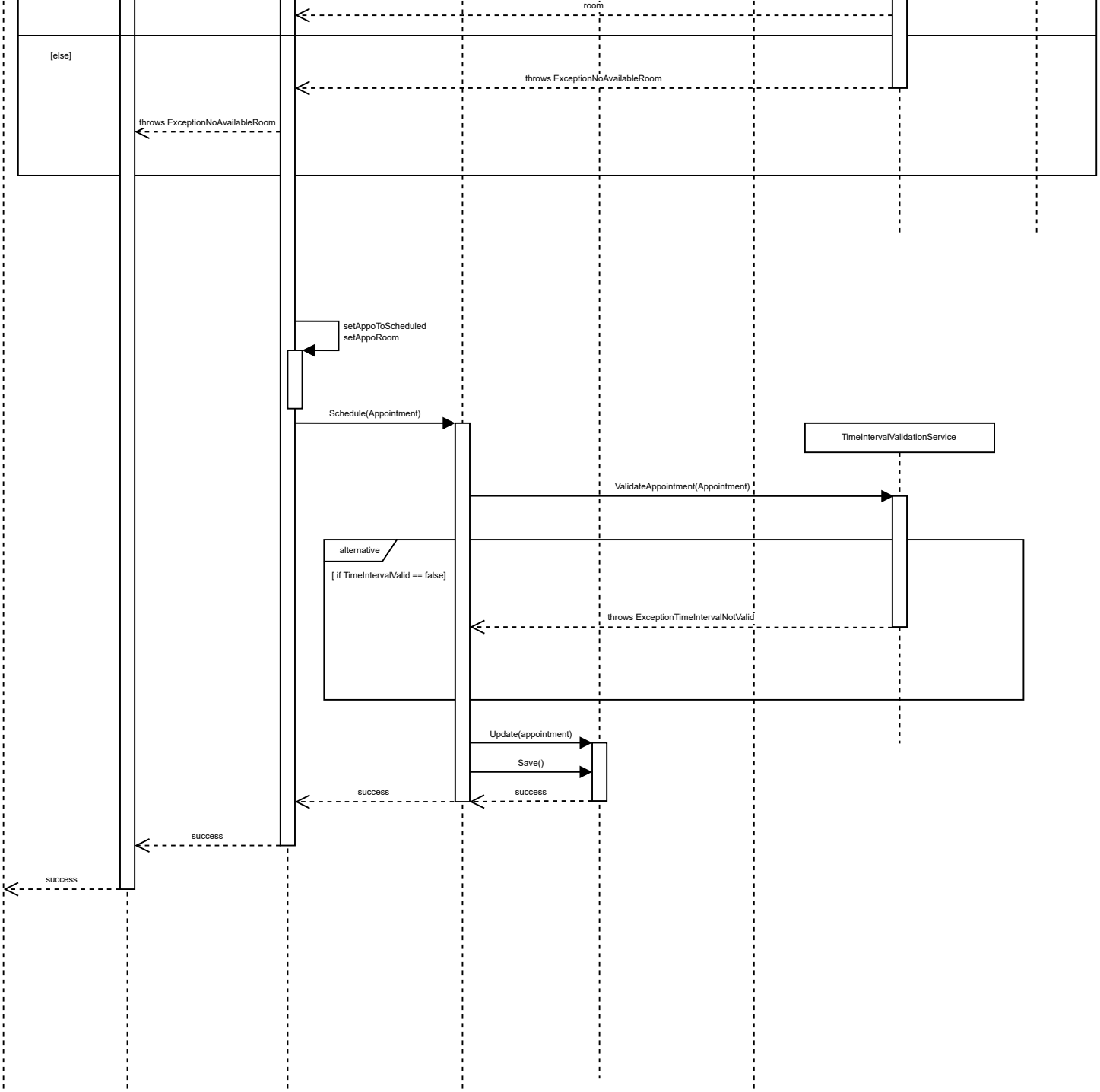
+ GetAllDoctorTakenIntervalsForDateExcept(int doctorId, DateTime date,int ignore) : Task<List<TimeInterval>>

+ GetAllRoomTakenIntervalsForDateExcept(int roomId, DateTime date,int ignore) : Task<List<TimeInterval>>


Frontend Application
ApplicationLayer
AppointmentService
AppointmentRepository
SchedulingAppointmentEvent
Click for start scheduling
CreateInitialAppointment()
CreateEmpty(Appointment app)
Add(appointment)
Save()
success
newAppointment
OK(newAppointment.Id)
SendEventStarted
<<create>>
AddEvent(started)
GetOne(appointmentId)
appointment
appointmentAddEventStarted
Save()
success
success
success
Click for picking date
Click for next
parallel
PickedDateForAppointment(date,appointmentId)
GetById(int AppointmentId)
GetById(int AppointmentId)
appointment
appointment
setDateAppo
Update(Appointment)
Update(appointment)
Save()
success
success
success
SendEventPickedDate
<<create>>
AddEvent(pickedDate)
GetOne(appointmentId)
appointment
appointmentAddEventDate
Save()


success
success
success
Click for picking speciality
Click for next
SendEventPickedSpeciality
<<create>>
AddEvent(pickedSpeciality)
GetOne(appointmentId)
appointment
appointmentAddEventPickedSpeciality
Save()
success
success
success
Click for picking doctor
Click for next
parallel
PickedDoctorForAppointment(docttorId,appointmentId)
GetById(int AppointmentId)
GetById(int AppointmentId)
appointment
DoctorService
DoctorRepository
appointment
GetById(int DoctorId)
GetById(int DoctorId)
doctor
doctor
setDoctorAppo
Update(Appointment)
Update(appointment)
Save()
success
success
success
SendEventPickedDoctor
<<create>>
AddEvent(pickedDoctor)
GetOne(appointmentId)
appointment
appointmentAddEventPickedDoctor


Save()
success
success
success
Click for picking time
Click for next
parallel
PickedTimeForAppointment(timeinterval, appointmentId)
GetById(int AppointmentId)
GetById(int AppointmentId)
appointment
appointment
setTimeAppo
Update(Appointment)
Update(appointment)
Save()
success
success
success
SendEventPickedTime
<<create>>
AddEvent(pickedTime)
GetOne(appointmentId)
appointment
appointmentAddEventTime
Save()
success
success
success
Click schedule
ScheduleAppointment(appointmentId)
DoctorService
DoctorRepository
GetById(int AppointmentId)
GetById(int AppointmentId)
appointment
appointment
GetFirstAvailableRoom(TimeInterval)
GetHospitalExaminationRoomes
Ienumerable<room> rooms
alternative
[FoundRoom != null]


room
[else]
throws ExceptionNoAvailableRoom
throws ExceptionNoAvailableRoom
setAppoToScheduled
setAppoRoom
Schedule(Appointment)
TimeIntervalValidationService
ValidateAppointment(Appointment)
alternative
[if TimeIntervalValid == false]
throws ExceptionTimeIntervalNotValid
Update(appointment)
Save()
success
success
success
success