AASS Semestral project

Lukáš Častven, Michal Kilian
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xcastven@stuba.sk, xkilian@stuba.sk

May 12, 2025

1 Application overview

Our application is a simple medical calendar in which a patient can request and appointment, doctor can accept it or deny it as well as add resources such as specialized medical facility, medicine, or equipment. In this calendar patient can also see his/her medical conditions and prescriptions history.

2 Stakeholders

Stakeholder	**Description**
Patient	Wants to book appointment with doctor.
Medical staff	Accepts or rejects appointment requests, and selects specialized facilities, equipment or medicine.
Other hospital staff	Needs to see the availability of other doctors and specialized rooms for their patients.
Hospital	Desires a new application to organize doctors and appointments.

Table 1: Stakeholders and their descriptions

3 Motivation

The following motivation diagram contains the drivers, assessments, goals, outcomes, and requirements related to the 3 identified stakeholders. Patient, Hospital, Nurse, Doctor.

The core problems, ”High operational costs”, ”Long waiting times”, and a ”Bad reputation for possible new hires”, drive the hospital for change. These drivers are associated with issues such as ”Ineffective allocation”, ”Medical staff frustration”, ”Unavailability of specialized rooms”, and patients' ”Waiting times.”

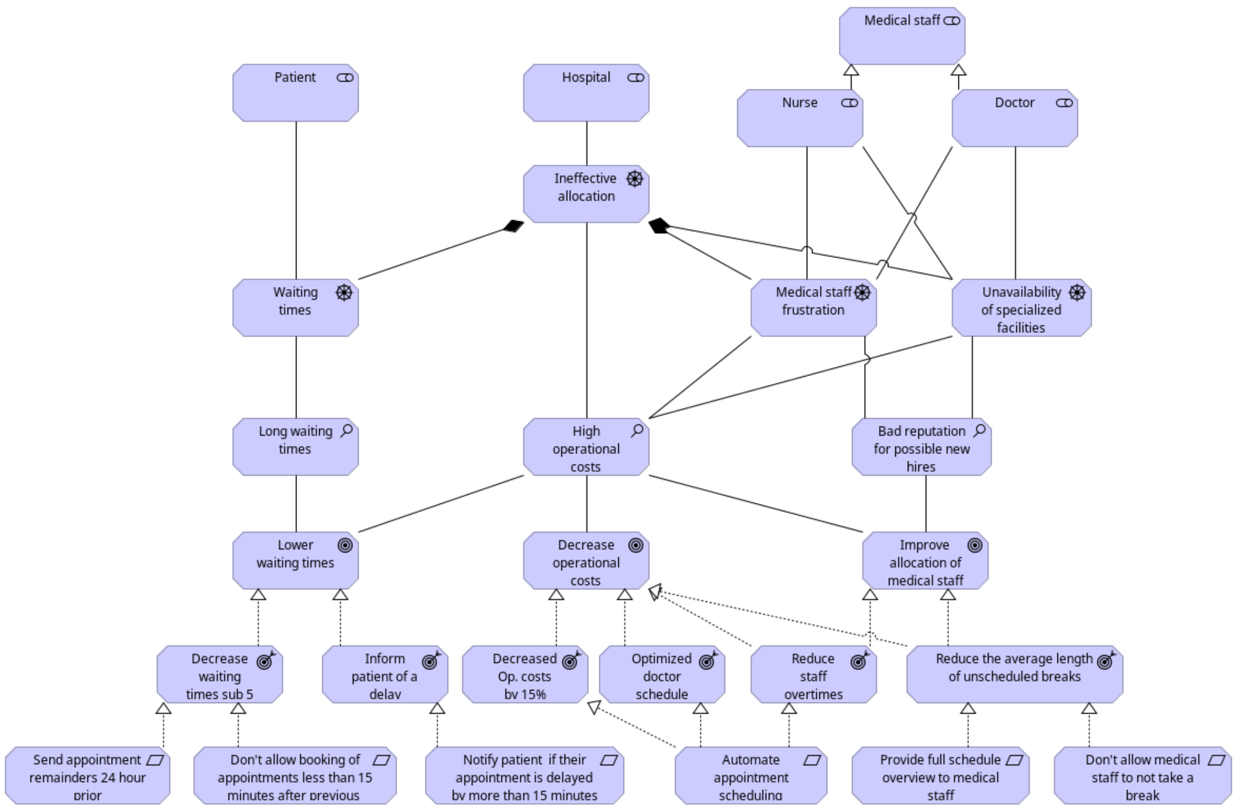

Medical staff
Patient
Hospital
Nurse
Doctor
Ineffective allocation
Waiting times
Medical staff frustration
Unavailability of specialized facilities
Long waiting times
High operational costs
Bad reputation for possible new hires
Lower waiting times
Decrease operational costs
Improve allocation of medical staff
Decrease waiting times sub 5
Inform patient of a delay
Decreased Op. costs by 15%
Optimized doctor schedule
Reduce staff overtimes
Reduce the average length of unscheduled breaks
Send appointment remainders 24 hour prior
Don't allow booking of appointments less than 15 minutes after previous
Notify patient if their appointment is delayed by more than 15 minutes
Automate appointment scheduling
Provide full schedule overview to medical staff
Don't allow medical staff to not take a break

Figure 1: Motivation layer

4 Value stream

The value stream of scheduling an appointment realizes the outcome of optimized doctor schedule.

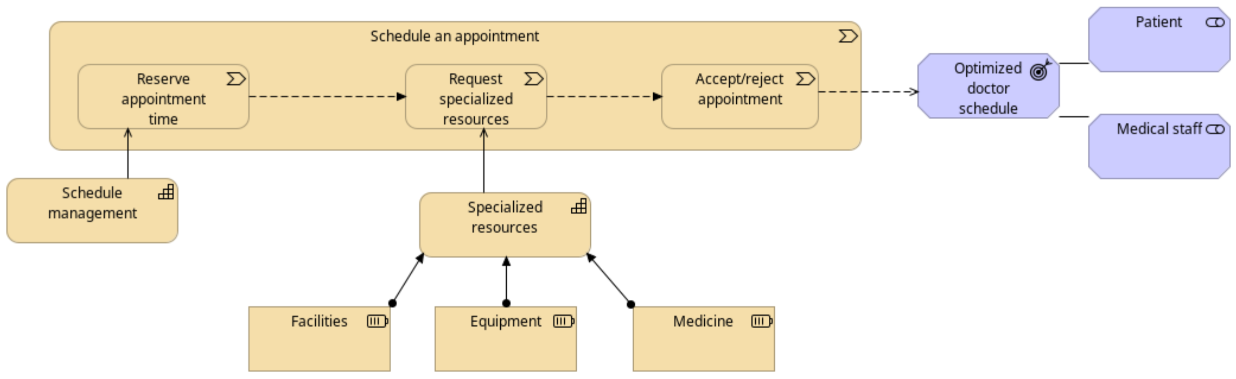


Figure 2: Value stream: Scheduling appointment

5 Business layer

Following diagram models three business processes, which serve the patient and medical staff to manage schedule.

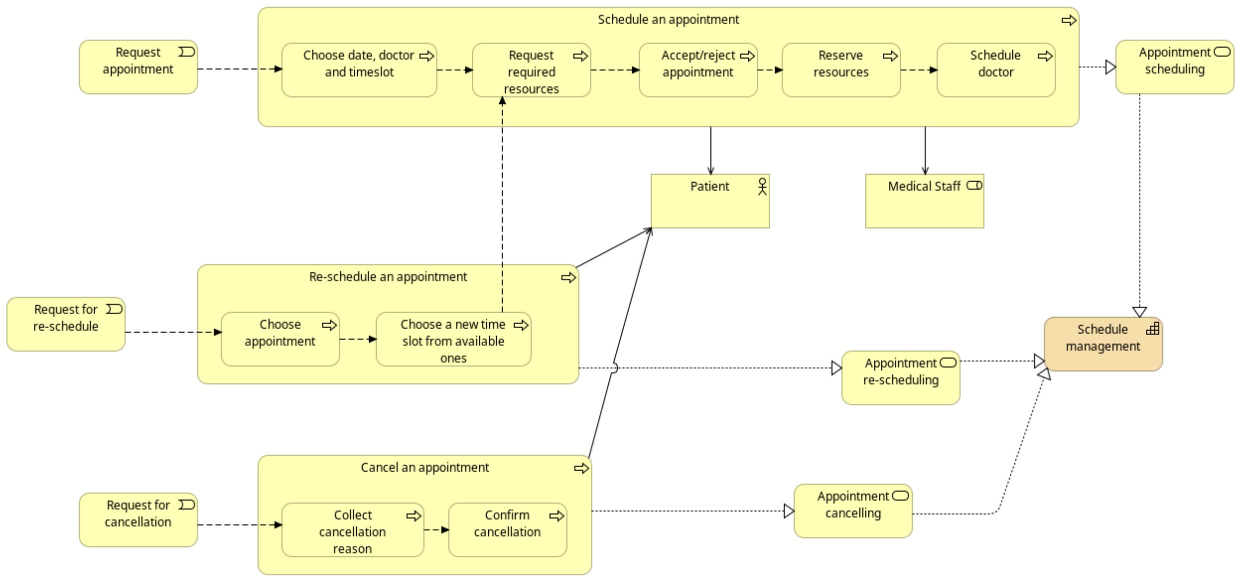


Figure 3: Business layer

6 Mockups

The patient interface, Figure 4, shows a design centered on appointment scheduling. Figure 5 illustrates the UI for the same use case, but with more control and additional resources requested by the medical staff.

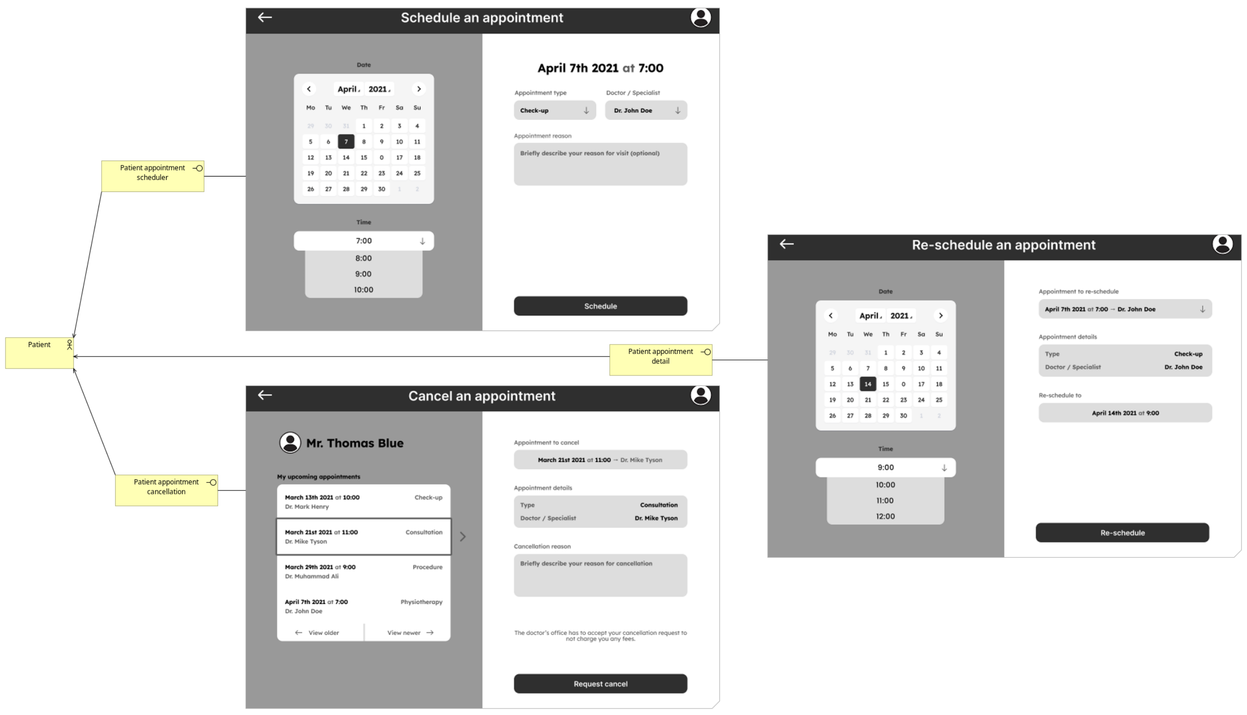
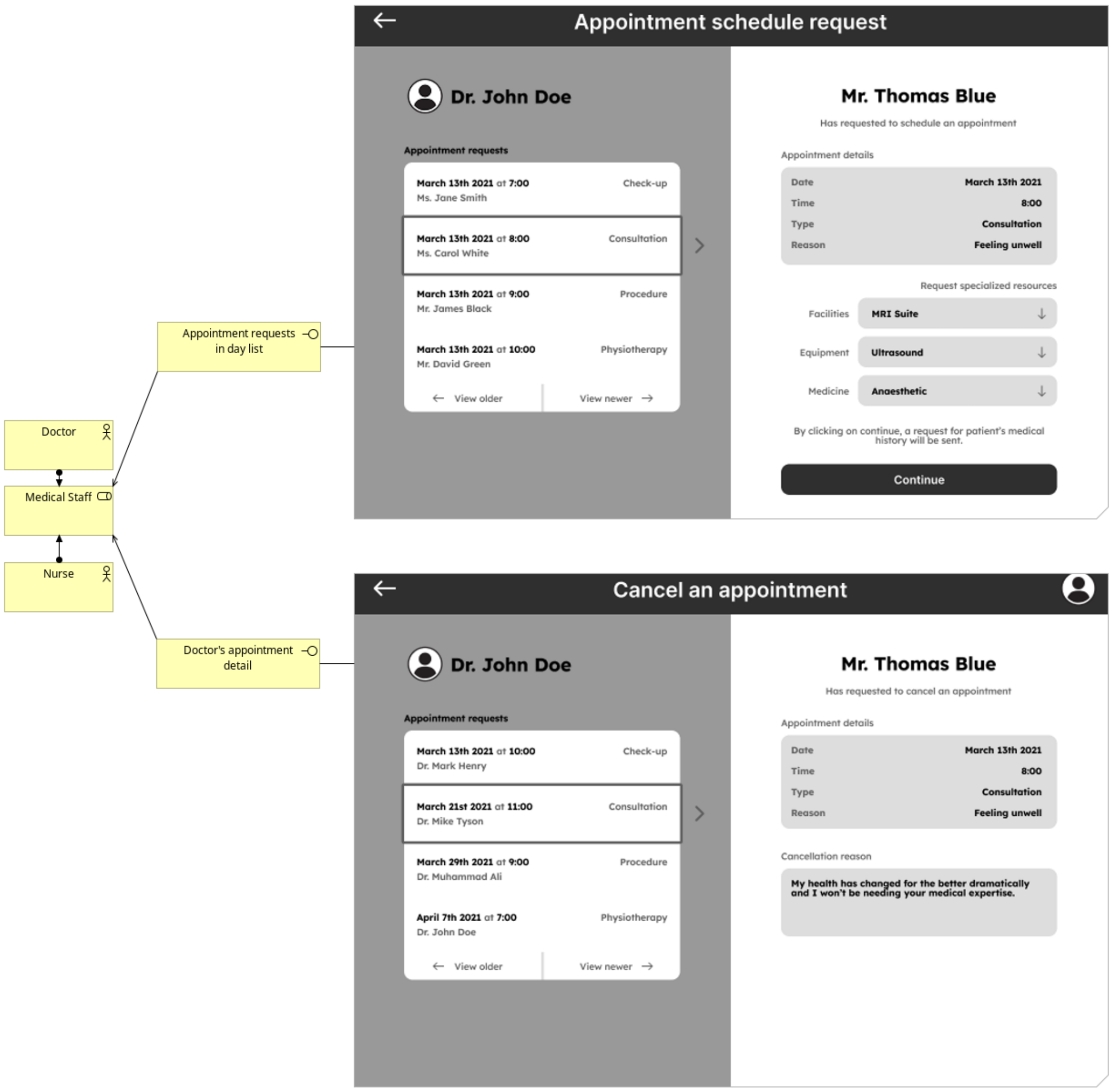

Patient
Patient appointment scheduler
Patient appointment detail
Patient appointment cancellation
Schedule an appointment
Date
April 2021
Time
7:00
8:00
9:00
10:00
April 7th 2021 at 7:00
Appointment type
Check-up
Doctor / Specialist
Dr. John Doe
Appointment reason
Briefly describe your reason for visit (optional)
Schedule
Re-schedule an appointment
Appointment to re-schedule
April 7th 2021 at 7:00 → Dr. John Doe
Appointment details
Type Check-up
Doctor / Specialist Dr. John Doe
Re-schedule to
April 14th 2021 at 9:00
Re-schedule
Cancel an appointment
Mr. Thomas Blue
My upcoming appointments
March 13th 2021 at 10:00 Dr. Mark Henry Check-up
March 21st 2021 at 11:00 Dr. Mike Tyson Consultation
March 29th 2021 at 9:00 Dr. Muhammad Ali Procedure
April 7th 2021 at 7:00 Dr. John Doe Physiotherapy
View older
View newer
Appointment to cancel
March 21st 2021 at 11:00 → Dr. Mike Tyson
Appointment details
Type Consultation
Doctor / Specialist Dr. Mike Tyson
Cancellation reason
Briefly describe your reason for cancellation
The doctor's office has to accept your cancellation request to not charge you any fees.
Request cancel

Figure 4: Patient's UI mockups



Appointment schedule request

Dr. John Doe

Appointment requests

March 13th 2021 at 7:00 Ms. Jane Smith	Check-up
March 13th 2021 at 8:00 Ms. Carol White	Consultation
March 13th 2021 at 9:00 Mr. James Black	Procedure
March 13th 2021 at 10:00 Mr. David Green	Physiotherapy

← View older | View newer →

Mr. Thomas Blue

Has requested to schedule an appointment

Appointment details

Date	March 13th 2021
Time	8:00
Type	Consultation
Reason	Feeling unwell

Request specialized resources

Facilities	MRI Suite
Equipment	Ultrasound
Medicine	Anaesthetic

By clicking on continue, a request for patient's medical history will be sent.

Continue

Cancel an appointment

Dr. John Doe

Appointment requests

March 13th 2021 at 10:00 Dr. Mark Henry	Check-up
March 21st 2021 at 11:00 Dr. Mike Tyson	Consultation
March 29th 2021 at 9:00 Dr. Muhammad Ali	Procedure
April 7th 2021 at 7:00 Dr. John Doe	Physiotherapy

← View older | View newer →

Mr. Thomas Blue

Has requested to cancel an appointment

Appointment details

Date	March 13th 2021
Time	8:00
Type	Consultation
Reason	Feeling unwell

Cancellation reason

My health has changed for the better dramatically and I won't be needing your medical expertise.

Figure 5: Medical staff's UI mockups

7 Architecture

We will implement a user interface and a back-end components to support the "Schedule an appointment" business process.

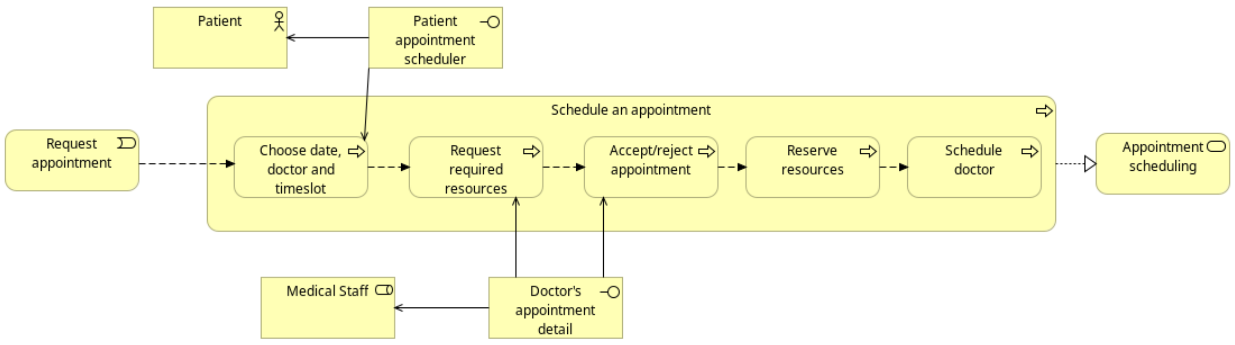

Patient
Patient appointment scheduler
Schedule an appointment
Request appointment
Choose date, doctor and timeslot
Request required resources
Accept/reject appointment
Reserve resources
Schedule doctor
Appointment scheduling
Medical Staff
Doctor's appointment detail

Figure 6: Schedule an appointment process implementation

The implementation will consist of a web application front-end developed using Stencil.js, and back-end written in 4 different architectures:

1. A monolithic backend
2. Microservices
3. Camunda oriented architecture
4. Event driven architecture with Kafka

The frontend implementation of the web app doesn't have to change/adapt to the different architectures and is written once.

How the web application is connected to the mock-ups and the process as a whole is shown in Figure 7. Both types of users access the core functionality of appointment scheduling through the Web app. The diagram further details that Medical Staff interact specifically with the "Doctor's appointment detail", while Patients utilize the "Patient appointment scheduler" interface within the Web app.

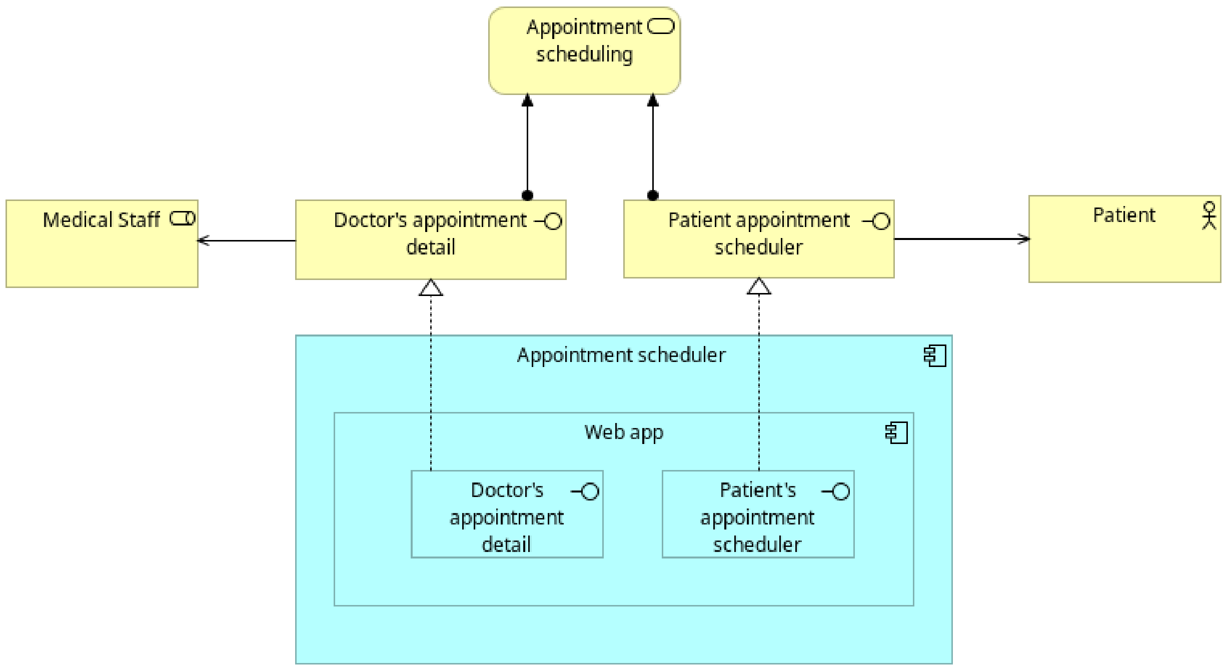

Appointment scheduling
Medical Staff
Doctor's appointment detail
Patient appointment scheduler
Patient
Appointment scheduler
Web app
Doctor's appointment detail
Patient's appointment scheduler

Figure 7: Business Application

7.1 A monolithic backend

As for the first implementation, Figure 8 details the internal structure and data dependencies of the core "Appointment scheduler" component. This component interacts with several data objects including "Users" (specialized into "Patients" and "Doctors"), "Free timeslots", "Appointments", and "Resources" (which are further categorized into "Facility", "Equipment", and "Medicine"). The diagram also highlights capabilities provided by this component, such as "List doctors", "List doctors timeslots", "Manage resources", and the main "Appointment scheduling" function itself.

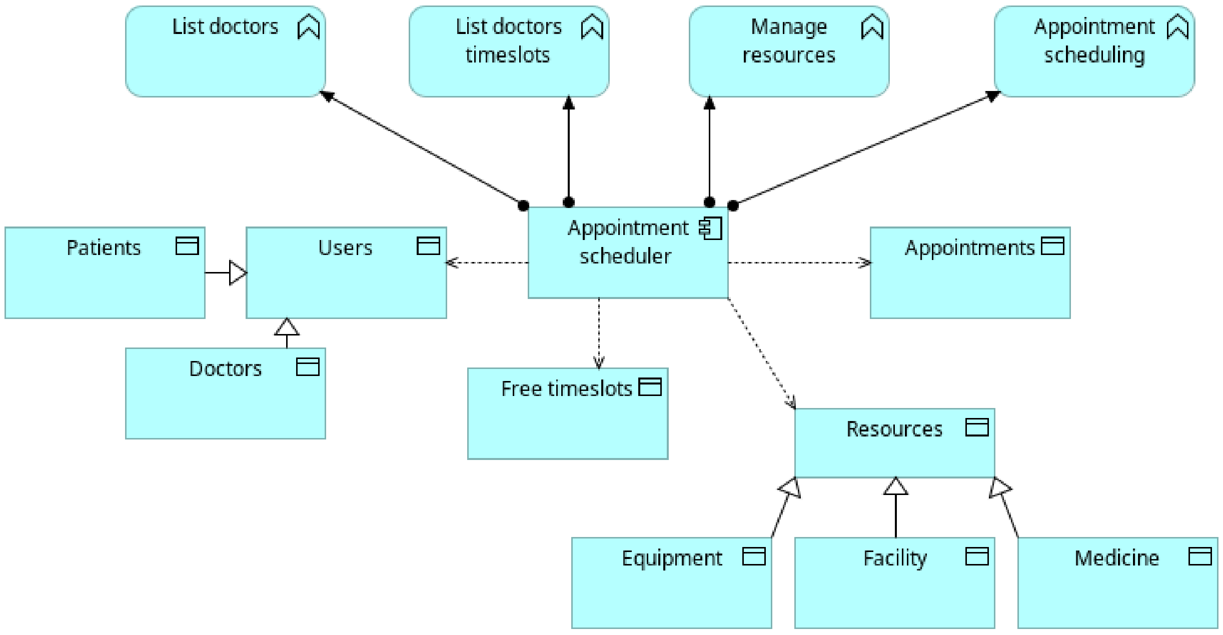


Figure 8: Application architecture pt. 1: A monolithic backend

7.2 Microservices

Figure 9 represents the second implementation and illustrates the flow of interactions between the "Web app", an "API Gateway", and various microservices. The "API Gateway" acts as an entry point, routing requests from the "Web app" to the appropriate microservices: "User service", "Appointment service", and "Resources service".

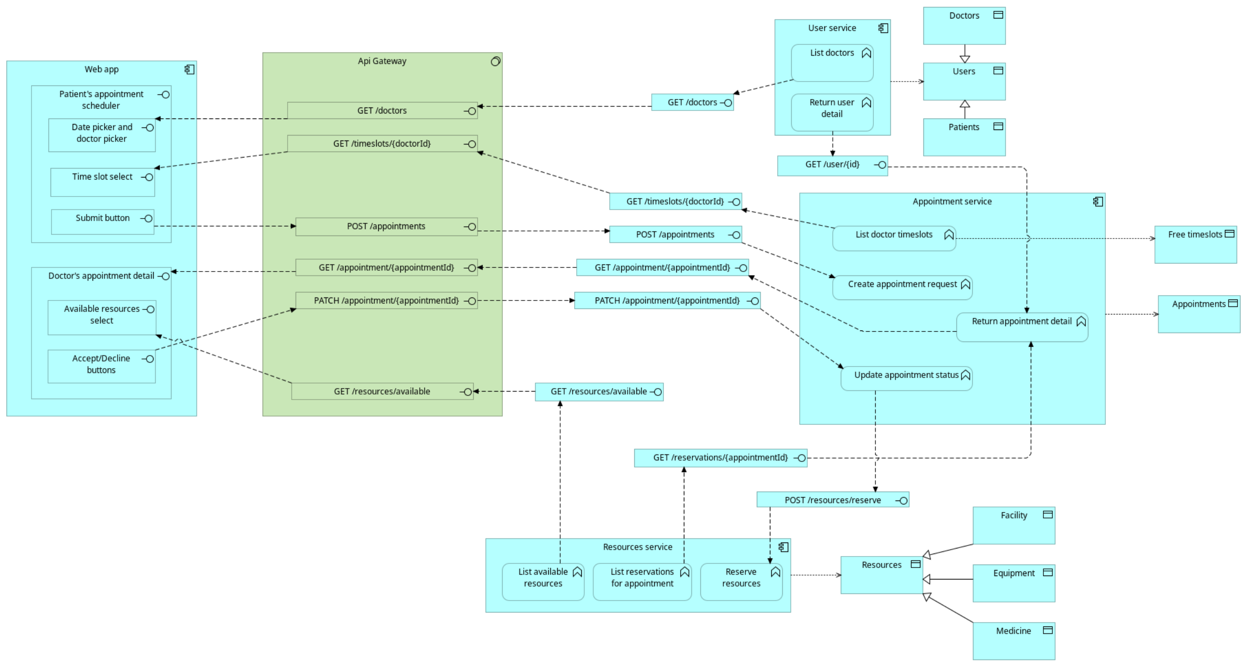

Web app
Patient's appointment scheduler
Date picker and doctor picker
Time slot select
Submit button
Doctor's appointment detail
Available resources select
Accept/Decline buttons
Api Gateway
GET /doctors
GET /timeslots/{doctorId}
POST /appointments
GET /appointment/{appointmentId}
PATCH /appointment/{appointmentId}
GET /resources/available
User service
List doctors
Return user detail
GET /doctors
GET /user/{id}
Doctors
Users
Patients
GET /timeslots/{doctorId}
POST /appointments
GET /appointment/{appointmentId}
PATCH /appointment/{appointmentId}
Appointment service
List doctor timeslots
Create appointment request
Return appointment detail
Update appointment status
Free timeslots
Appointments
GET /resources/available
GET /reservations/{appointmentId}
POST /resources/reserve
Resources service
List available resources
List reservations for appointment
Reserve resources
Resources
Facility
Equipment
Medicine

Figure 9: Application architecture pt. 2: Microservices

7.3 Camunda oriented architecture

Figure 10 models how the process can be supported with the integration of Camunda, specifically showing the "Camunda Schedule appointment process" component. This process interacts with an "External worker service" to "Reserve resources". Since our implementation doesn't use embedded Camunda, it can slightly resemble event driven architecture, since Camunda emits topics, which the external worker consumes and then submits the task results back to Camunda.

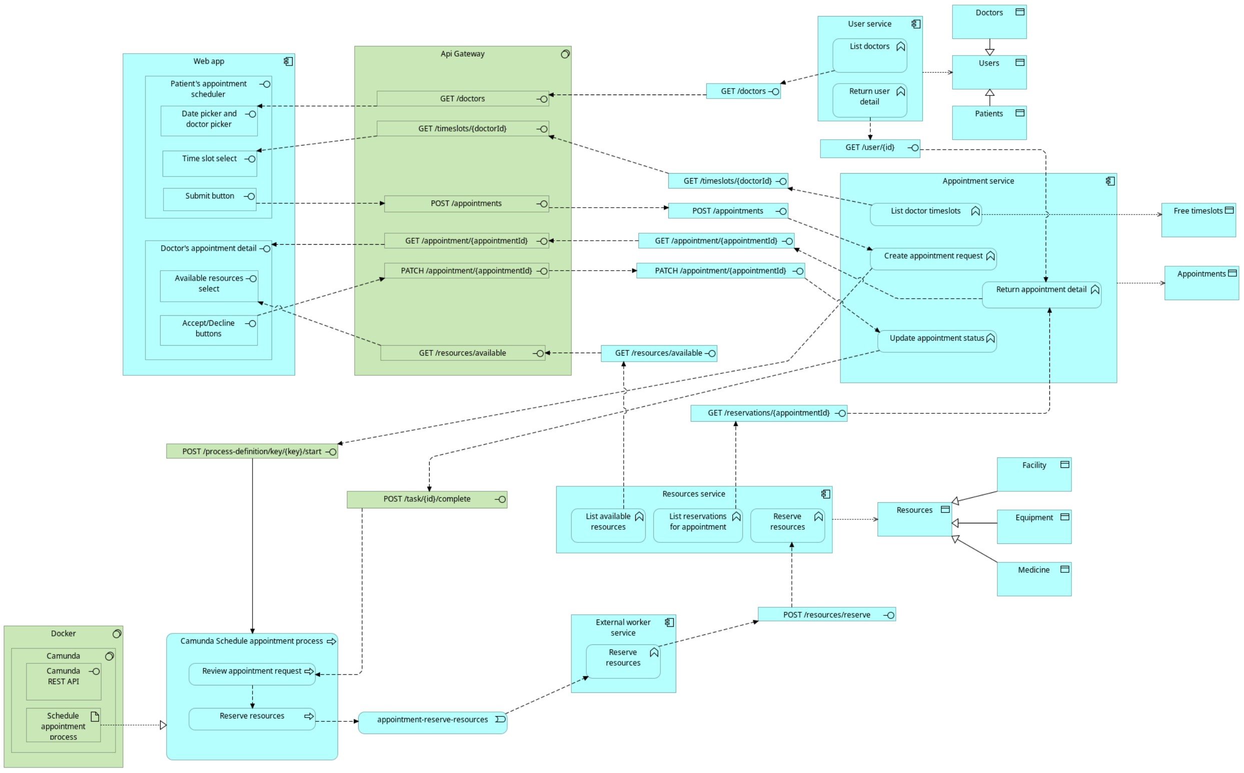

Web app
Patient's appointment scheduler
Date picker and doctor picker
Time slot select
Submit button
Doctor's appointment detail
Available resources select
Accept/Decline buttons
Api Gateway
GET /doctors
GET /timeslots/{doctorId}
POST /appointments
GET /appointment/{appointmentId}
PATCH /appointment/{appointmentId}
GET /resources/available
User service
List doctors
Return user detail
GET /doctors
GET /user/{id}
Doctors
Users
Patients
GET /timeslots/{doctorId}
POST /appointments
GET /appointment/{appointmentId}
PATCH /appointment/{appointmentId}
GET /resources/available
Appointment service
List doctor timeslots
Create appointment request
Return appointment detail
Update appointment status
Free timeslots
Appointments
GET /reservations/{appointmentId}
POST /process-definition/key/{key}/start
POST /task/{id}/complete
Resources service
List available resources
List reservations for appointment
Reserve resources
Resources
Facility
Equipment
Medicine
Docker
Camunda
Camunda REST API
Schedule appointment process
Camunda Schedule appointment process
Review appointment request
Reserve resources
appointment-reserve-resources
External worker service
Reserve resources
POST /resources/reserve

Figure 10: Application architecture pt. 3: Camunda oriented architecture

Figure 11 presents a business process model diagram detailing the "Camunda Schedule appointment process". The process begins with an "Appointment request". This triggers a user task, "Review appointment request", where a decision is made. An exclusive gateway, labeled "Decision made?", routes the flow. If the request is "Accepted", a service task "Reserve resources" is executed, leading to the "Appointment scheduled" end event. If "Denied", the process directly reaches the "Appointment denied" end event.

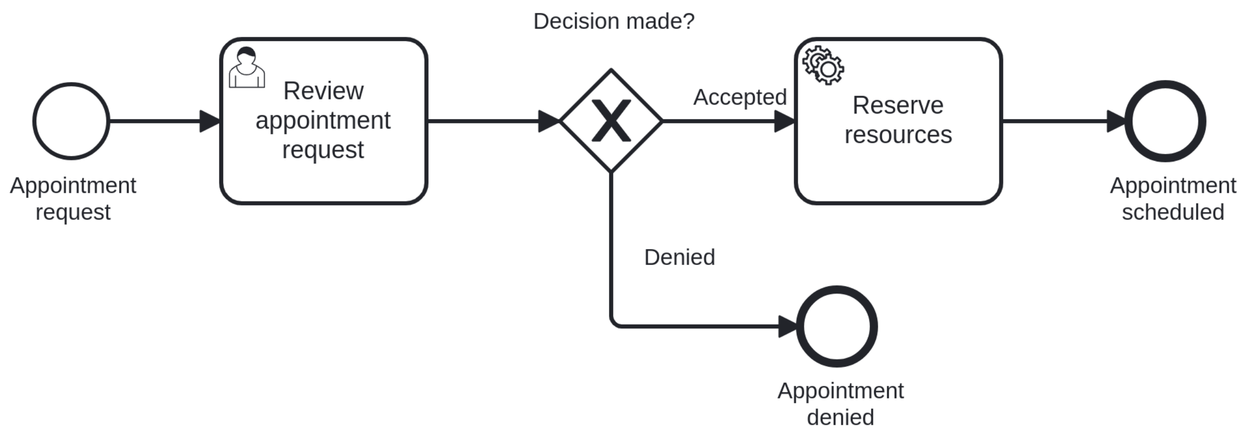


Figure 11: Schedule appointment process BPMN

7.4 Event driven architecture with Kafka

Figure 12 details the usage of Kafka for asynchronous communication within the system. Kafka is run in a Docker container, which includes a ”Kafka Custom TCP listener” endpoint. The ”Appointment service” publishes ”Appointment status update” messages to Kafka, and the ”Resources service” consumes these messages and publishes ”Resource reserved” messages.

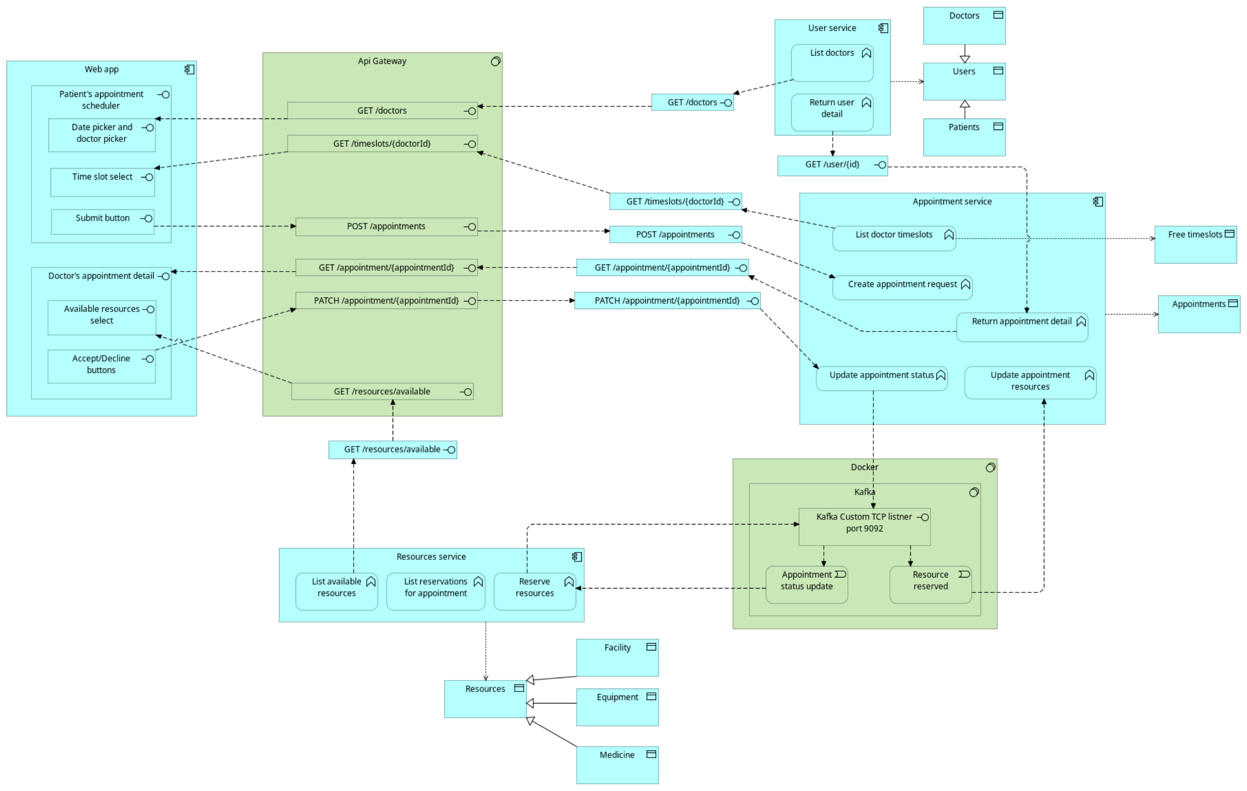

Web app
Patient's appointment scheduler
Date picker and doctor picker
Time slot select
Submit button
Doctor's appointment detail
Available resources select
Accept/Decline buttons
Api Gateway
GET /doctors
GET /timeslots/{doctorId}
POST /appointments
GET /appointment/{appointmentId}
PATCH /appointment/{appointmentId}
GET /resources/available
User service
List doctors
Return user detail
GET /user/{id}
Doctors
Users
Patients
GET /doctors
GET /timeslots/{doctorId}
POST /appointments
GET /appointment/{appointmentId}
PATCH /appointment/{appointmentId}
Appointment service
List doctor timeslots
Create appointment request
Return appointment detail
Update appointment status
Update appointment resources
Free timeslots
Appointments
GET /resources/available
Docker
Kafka
Kafka Custom TCP listner port 9092
Appointment status update
Resource reserved
Resources service
List available resources
List reservations for appointment
Reserve resources
Resources
Facility
Equipment
Medicine

Figure 12: Event driven architecture with Kafka

8 Conclusion

We successfully developed a simple medical calendar application in 4 different architectures, focusing on the core "Schedule an appointment" process.

We explored various architectural approaches for the backend. Implementing and modeling monolithic, microservices, Camunda oriented, and event-driven architectures using Kafka, without requiring frontend implementation to change.