# |. Partitionierung

#### Aufgabenorientierte Partitionierung

- User Management Service
- Organization Service (Doctors, Offices, etc.)
- Appointment Scheduling Service (+ Calender Data)
- Al Contact Agent Service (Fast Api)
- Notification Service
- Frontend Mobile App (Flutter)

# Application

- Exception
- Model
- Port
  - Incoming port
  - outgoing port

### Config

- Security

#### Domain

- Validation
- Service

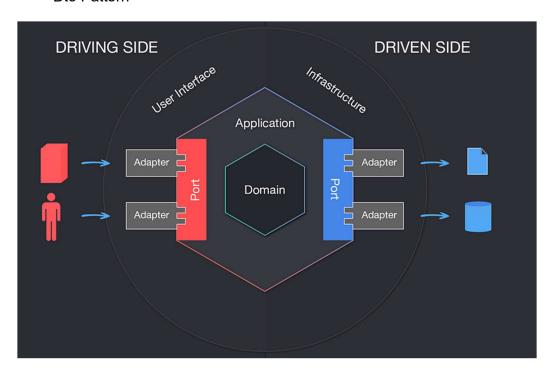
#### Infrastructure

- Incoming
  - Mapper
  - Rest
  - Events Reader
- Outgoing
  - Mapper
  - Event Sender
  - Database

#### Test

### II. Modularisierung

- SOLID
- III. Nutzung bewährter Entwurfsmuster
- Adapter Pattern
- Chain of Responsibility Pattern
- Iterator Pattern
- -Dto Pattern



#### IV. Wiederverwendbarkeit

- Adapter Pattern mit dem Fokus auf Domain
- Jeder Service kann durch mehrere apps benutzt werden (zb Notifikation Service)
- V. Nutzung von Vorhandenem und Verfügbarem
  - PostgreSQL
  - Twilio or Vonage (phone oder SMS calls)
  - Flutter Packages (UI)
  - Al model APIs
  - Java Dependency

### VI. Aspektorientierung

- Logging
- Security/auth checks
- Rate limiting

# VII. Ästhetik

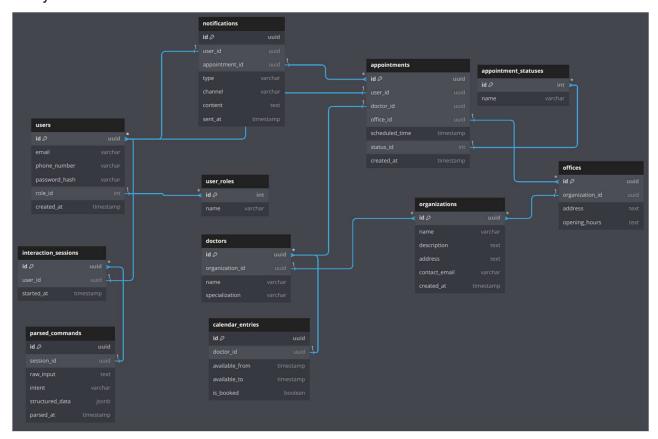
- API design: ( /api/v1/appointments)
- naming: AppointmentController, NotificationService, etc.
- Flutter UI: (Responsive, etc)
- Backend: Swagger/OpenAPI für Documentation von API

## DDD:

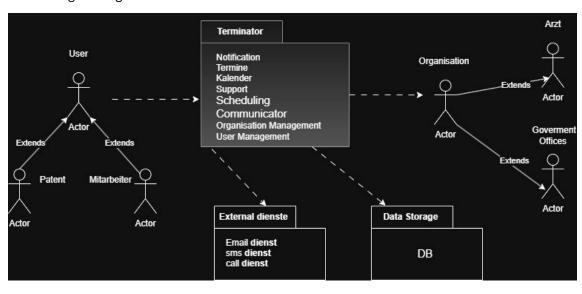
Domain: Terminator



## Entity:



#### Kontextabgrenzung



Bausteine