SDMA Technical Manual
Eine webbasierte Anwendung zur Verwaltung gemeinsamer Entscheidungen bei der Behandlung von Schizophrenie

Inhalt

1. Architektur

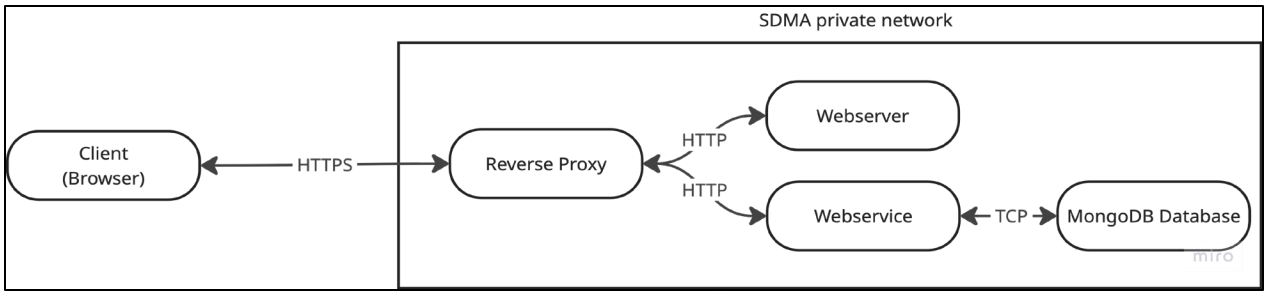
1.1. Technologien

Die SDMA-App ist als Webanwendung implementiert. Dies bietet die größte Flexibilität hinsichtlich der Geräte, auf denen die App genutzt werden kann. Auf der Frontend-Seite müssen die Datenplots zu Antipsychotika angezeigt werden, die sich interaktiv durch Benutzereingaben verändern. Um dies zu erreichen, wird React eingesetzt, da es die Entwicklung reaktiver Webanwendungen ermöglicht. Ein zusätzlicher Vorteil ist, dass der Code bei Bedarf in Zukunft leicht zu React Native migriert werden kann, falls eine native App erforderlich sein sollte.
Der Backend-Server ist mit Spring in Kotlin implementiert. Spring bietet eine schnelle und einfache Einrichtung für einen REST-Service und ist ein etabliertes Framework mit guter Unterstützung sowie einer Vielzahl ausgereifter Bibliotheken. Insbesondere die Spring-Security-Bibliothek ist sehr hilfreich bei der Implementierung der Authentifizierungs- und Autorisierungsprozesse, die die SDMA-App benötigt. Kotlin wird gegenüber Java gewählt, da es explizite Datenklassen, eine bessere Handhabung von Null-Werten und meiner Meinung nach einer besser lesbaren Syntax bietet.
Als Datenbank wird MongoDB verwendet, da sie eine sehr gute Anbindung an Spring bietet, einschließlich automatisch generierter Query-Methoden. Die von uns gespeicherten Daten passen zudem besser zu einer objektorientierten Datenbank, was im nächsten Abschnitt näher erläutert wird.

Das SDMA-Backend besteht aus vier Komponenten:

1. Reverse Proxy: Dies ist der Punkt, über den die gesamte Kommunikation von Clients läuft. Er ist dafür verantwortlich, sicherzustellen, dass die gesamte ausgehende Kommunikation über HTTPS verschlüsselt ist.
2. Webserver: Der Server, der die statischen Inhalte der Website ausliefert.
3. Webservice: Der Dienst, der für dynamische Inhalte zuständig ist, wie z. B. das Speichern und Abrufen von Benutzerdaten.

4. Datenbank: Die Datenbank, in der Benutzer- und medizinische Daten gespeichert werden.



1.2. *Erstellung und Bereitstellung*

Das SDMA-Backend wird mit Docker-Containern gebaut. Eine Einführung in Docker findet sich unter https://www.docker.com/resources/what-container/. Alle Komponenten sind plattformunabhängig und können auf allen Systemen gebaut und deployed werden, die Linux-Docker-Container ausführen können. Jede Komponente läuft in ihrem eigenen Container; diese Container werden über Docker Compose orchestriert. Die Docker-Compose-Konfiguration ist dafür verantwortlich, dass die Container über HTTP miteinander kommunizieren können, indem sie im selben (virtuellen) privaten Netzwerk platziert werden.

Dieses Repository enthält **vier verschiedene docker-compose-Konfigurationen** für unterschiedliche Umgebungen:

docker-compose-local.yml

Nützlich für die lokale Entwicklung. Der Webserver läuft hier in einem Development-Build, der Hot Reloading ermöglicht.

docker-compose.yml

Dies ist der einfachste Weg, die App zu deployen. Dabei werden die Docker-Images gebaut; hierfür ist Zugriff auf DockerHub notwendig, um Standard-Images zu laden.

docker-compose-production.yml

Dies ist die Konfiguration, die zur Orchestrierung des SDMA-Backends auf den MRI-Servern verwendet wird. Sie kommt zum Einsatz, wenn kein Zugriff auf DockerHub verfügbar ist. Alle benötigten Images müssen auf das Deployment-Ziel importiert werden. Siehe dazu die Dateien **docker-images/load_images.sh und docker-**

images/store_images.sh, die beschreiben, wie Images auf einer Maschine mit DockerHub-Zugriff gebaut, gezippt und anschließend auf das Deployment-Ziel importiert werden. Neben den selbst gebauten Images müssen auch die Standard-Images nginx:stable-alpine und mongo übertragen werden.

aws-compose.yml

Diese Konfiguration wurde für das Deployment auf AWS verwendet. Die Images müssen zuerst gebaut und in die ECR hochgeladen werden (siehe aws_deploy.sh), danach können sie in der docker-compose-Konfiguration referenziert werden.

In allen Konfigurationen muss unter services.nginx.volumes der Ordner mit allen notwendigen SSL-Dateien nach /etc/nginx/ssl gemountet werden. Die in diesem Repository enthaltenen SSL-Dateien sind selbstsignierte Zertifikate und dienen ausschließlich zu Testzwecken. Private SSL-Zertifikate für den Produktivbetrieb dürfen niemals geteilt werden und sollten auf dem Deployment-Ziel verbleiben sowie wie in docker-compose-production.yml eingebunden werden.

Die Daten werden an dem Speicherort abgelegt, der über services.mongo_db.volumes gemountet ist. Der Service kann anschließend im Hintergrund gestartet werden mit:

docker-compose -f <path to docker-compose configuration> up -d

Um das SDMA-Backend beim Systemstart zu starten und bei Abstürzen automatisch neu zu starten, sollte ein plattformspezifischer Service (z. B. systemd) verwendet werden. Alle Docker-Images können gebaut werden mit: **docker-compose -f docker-compose.yml build**

Build Requirements

- Der SDMA-Server ist betriebssystemunabhängig und kann auf allen Systemen deployed werden, die Linux-Docker-Container ausführen können
- Docker oder eine Alternative (Podman, RancherDesktop, Colima)
- Zugriff auf DockerHub

Deployment Requirements

- Docker oder eine Alternative (Podman, RancherDesktop, Colima)

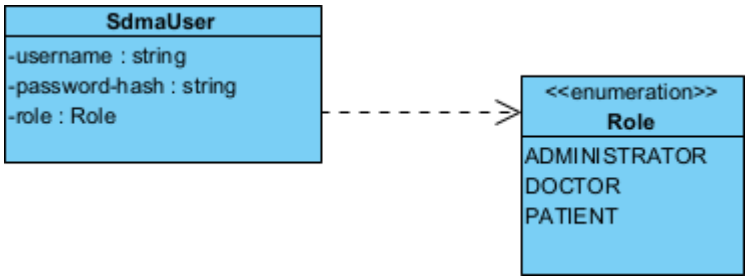
- Docker-Compose
- Zugriff auf DockerHub ODER importierte Images

2. Datenmodell

2.1. Benutzer

Um eine Authentifizierung und Autorisierung auf Basis der drei Rollen Administrator, Doctor und Patient zu implementieren, müssen alle Benutzer gespeichert werden. Dieses Datenmodell wird getrennt von dem Datenmodell gehalten, das die Business-Daten enthält. Für jeden Benutzer werden ein Benutzername, ein bcrypt-Hash des Passworts sowie die jeweilige Rolle gespeichert.

Benutzerdaten für Authentifizierung und Autorisierung werden somit getrennt von den Business-Daten verwaltet. Für jeden Benutzer werden ein Benutzername, ein bcrypt-Hash des Passworts und die zugehörige Rolle gespeichert.



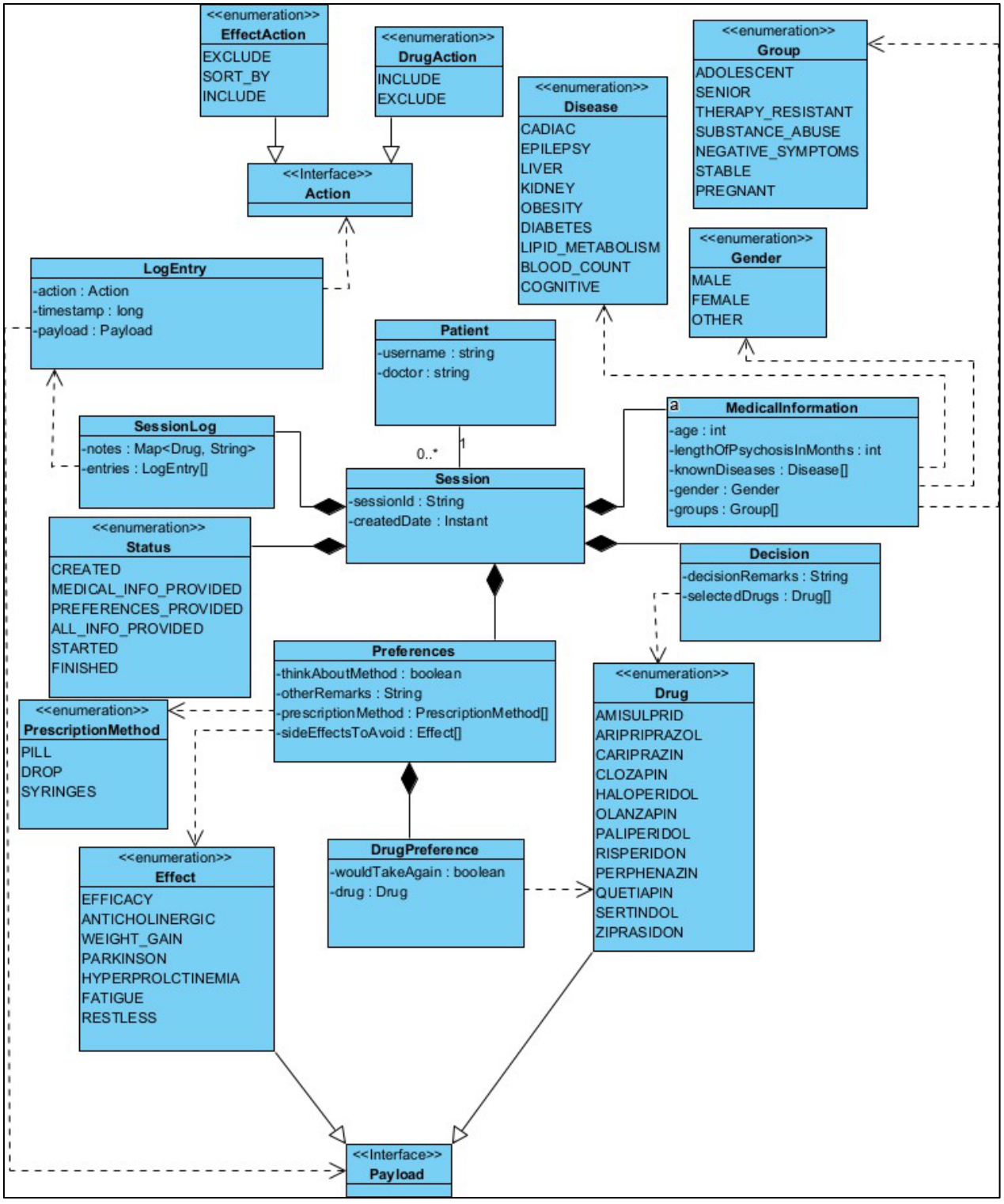
2.2. Medizinische Daten

Die Daten, mit denen gearbeitet wird, sind Patients und Sessions. Für jeden Patienten wird der behandelnde Arzt gespeichert. Eine Session besteht aus allen während der Sitzung bereitgestellten Daten, einschließlich der Eingaben in die in den folgenden Abschnitten beschriebenen Formulare. Für jede Frage sind die vordefinierten Antwortmöglichkeiten in einem Enum definiert.

Darüber hinaus wird das Log einer Session gespeichert, das aus allen Aktionen besteht,

die die Benutzer auf der interaktiven Benutzeroberfläche durchgeführt haben. Das Speichern dieses Logs ermöglicht es, eine Session zu sichern und sie an der Stelle wiederherzustellen, an der sie verlassen wurde. Zusätzlich bietet es die Möglichkeit zu analysieren, wie intensiv ein Patient mit der Oberfläche interagiert hat und welche Aktionen er durchgeführt hat.

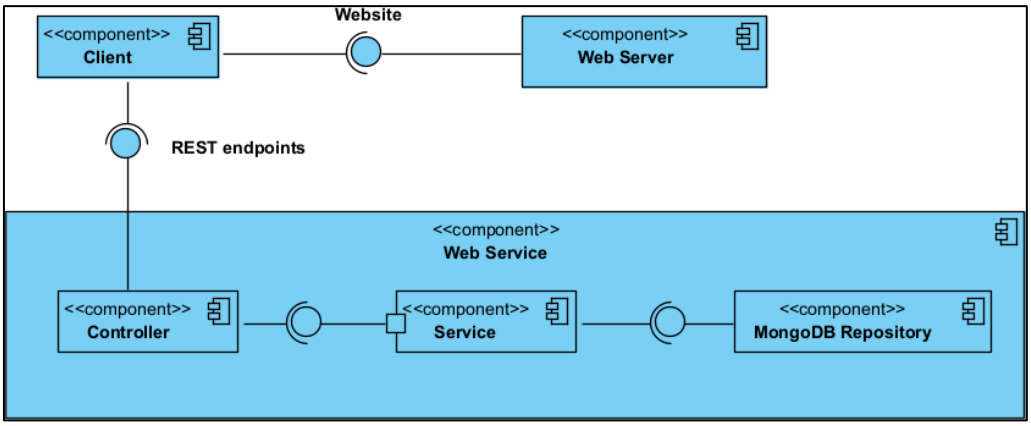
Ein Session-Log besteht aus Aktionen auf Effects und Antipsychotics. Jede Aktion besteht aus einem Typ, der einer möglichen Interaktion in der interaktiven Benutzeroberfläche entspricht, einem Zeitstempel und dem Payload, also dem jeweiligen Effekt bzw. Antipsychotikum. Alle während einer Session gesammelten Daten werden in einem Session-Objekt gespeichert.



Darstellung 1: Datenmodel

2.3 Komponentenübersicht und Interaktion

Die SDMA-Applikation ist in drei Hauptkomponenten unterteilt. Der Webserver hostet den kompilierten React-Code als clientseitig gerenderte Webanwendung. Die Spring-Backend-Komponente stellt REST-Endpoints bereit, um den Login durchzuführen sowie Daten zu senden und abzurufen. Die Pfade zu den REST-Endpoints sind in der Webseite definiert und werden daher direkt vom Browser des Clients aufgerufen. Die Datenbank-Komponente wird ausschließlich vom Spring-Service genutzt, um Daten zu lesen oder zu persistieren.



Darstellung 2: Komponentenübersicht

Die Architektur der REST-Backend-Komponente folgt der typischen Struktur, die bei Spring-Anwendungen verwendet wird. Der Service nutzt zwei getrennte Datenmodelle. Das Data-Transfer-Model wird für die REST-Kommunikation nach außen verwendet, während das Database-Model intern genutzt wird. Für Objekte, die identisch sind, wird das Database-Model auch für die externe Kommunikation verwendet. Sollten sich die Anforderungen an die Datenmodelle ändern, kann problemlos ein Konverter ergänzt werden.

Eine REST-Anfrage trifft zunächst in der REST-Controller-Schicht ein. Dort wird die Anfrage autorisiert und – falls notwendig – in das interne Datenmodell konvertiert. Anschließend wird die entsprechende Service-Methode aufgerufen. Die Service-Schicht führt die Business-Logik aus und speichert oder liest Daten über das Repository.

Durch die Verwendung der Spring Data MongoDB Repositories können Repository-

Methoden automatisch generiert werden, ohne dass Datenbankabfragen in der nativen MongoDB-Sprache geschrieben werden müssen. Dies ist insbesondere geeignet für diese Anwendung, da lediglich einfache save- und find-by-Operationen benötigt werden.

3. Schnittstellendokumentation

Die SDMA-App verfügt über keine externen Schnittstellen; die gesamte Interaktion erfolgt über die Website-Benutzeroberfläche. Im Folgenden finden Sie eine Liste aller internen Schnittstellen, die von der Website verwendet werden.

PUT changePassword

PUT /user

> Body Parameters


```
{
"username": "string", "password": "string"
}
```


Params

Name	Location	Type	Required	Description
body	body	CreateUserDto	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

GET patients

GET /user/patients

> Response Examples

> 200 Response

[{"username":"string","doctor":"string"}]

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[Patient]	false	none		none
» username	string	false	none		none
» doctor	string	false	none		none

PUT changePatientDoctor

PUT /user/patients

> Body Parameters

```
{
"username": "string", "password": "string", "doctor": "string"
}
```

Params

Name	Location	Type	Required	Description
body	body	CreatePatientDto	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

POST createPatient

POST /user/patients

> Body Parameters


```
{
"username": "string", "password": "string", "doctor": "string"
}
```


Params

Name	Location	Type	Required	Description
body	body	CreatePatientDto	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

POST createDoctor

POST /user/doctors Body Parameters


```
{
"username": "string", "password": "string"
}
```


Params

Name	Location	Type	Required	Description
body	body	CreateUserDto	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

POST createAdministrator

POST /user/administrators

> Body Parameters


```
{
"username": "string", "password": "string"
}
```


Params

Name	Location	Type	Required	Description
body	body	CreateUserDto	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
201	Created	Created	None

GET findUser

GET /user/{username}

Params

Name	Location	Type	Required	Description
username	path	string	yes	none

> Response Examples

> 200 Response

{"username":"string","role":"ADMINISTRATOR"}

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	SdmaUserDto

DELETE deleteUser

DELETE /user/{username}

Params

Name	Location	Type	Required	Description
username	path	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

GET usersByRole

GET /user/role/{role}

Params

Name	Location	Type	Required	Description
role	path	string	yes	none

Enum

Name	Value
role	ADMINISTRATOR
role	DOCTOR
role	PATIENT

Response Examples

200 Response

[{"username":"string","role":"ADMINISTRATOR"}]

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[SdmaUserDto]	false	none		none
» username	string	false	none		none
» role	string	false	none		none

Enum

Name	Value

role	ADMINISTRATOR

Name	Value
role	DOCTOR
role	PATIENT

GET currentUser

GET /user/current

> Response Examples

> 200 Response

{"username":"string","role":"ADMINISTRATOR"}

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	SdmaUserDto

POST username

POST /authenticate

> Response Examples

> 200 Response

"string"

Responses

HTTP Status Code	Meaning	Description	Data schema

200	OK	OK	string

PUT providePatientPreferences

PUT /doctor/session/{session-id}/preferences

> Body Parameters


```
{
"goals": [ "string"
],
"drugsTakenBefore": [
{
"drug": "AMISULPRID",
"wouldTakeAgain": true
}
],
"otherDrugsTaken": [ "string"
],
"sideEffectsToAvoid": [ "EFFICACY"
],
"otherSideEffectsToAvoid": [ "string"
],
"thinkAboutMethod": true, "methodPreferences": [
"PILL"
],
"otherRemarks": [ "string"
]
}
```


Params

Name	Location	Type	Required	Description
session-id	path	string	yes	none
body	body	Preferences	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

GET getLastMedicalInformation

GET /doctor/session/{session-id}/medical_info

Params

Name	Location	Type	Required	Description
session-id	path	string	yes	none

> Response Examples 200 Response

> {"age":0,"sex":"MALE","lengthOfPsychosisInMonths":0,"knownDiseases":["CARDIAC"],"otherDiseases

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	MedicalInformation

PUT provideMedicalInformation

PUT /doctor/session/{session-id}/medical_info

> Body Parameters


```
{
"age": 0, "sex": "MALE",
"lengthOfPsychosisInMonths": 0,
"knownDiseases": [ "CARDIAC"
],
"otherDiseases": [ "string"
],
"groups": [ "ADOLESCENT"
]
```

}

Params

Name	Location	Type	Required	Description
session-id	path	string	yes	none
body	body	MedicalInformation	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

POST takeLastMedicalInformation

POST /doctor/session/{session-id}/medical_info

Params

Name	Location	Type	Required	Description
session-id	path	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

PUT saveSessionProgress

PUT /doctor/session/{session-id}/log

Body Parameters

{
"actions": [
{

```
"type": "string", "timestamp": 0, "payload": "string"
}
],
"notes": {
"property1": { "commonReasons": [
"BAD_EFFICACY"
],
"otherReasons": "string"
},
"property2": { "commonReasons": [
"BAD_EFFICACY"
],
"otherReasons": "string"
}
}
}
```


Params

Name	Location	Type	Required	Description
session-id	path	string	yes	none
body	body	SessionLogDto	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

PUT saveSessionDecision

PUT /doctor/session/{session-id}/decision Body Parameters


```
{
"selectedDrugs": [ "AMISULPRID"
],
"decisionRemarks": "string"
```

}

Params

Name	Location	Type	Required	Description
session-id	path	string	yes	none
body	body	Decision	no	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

POST createSession

POST /doctor/patients/{patient-name}/session

Params

Name	Location	Type	Required	Description
patient-name	path	string	yes	none

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	None

GET findActiveSession

GET /doctor/session

> Response Examples 200 Response

{"sessionId":"string","status":"string","created":0,"lastModified":0}

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	SessionDto

GET getSessionDetails

GET /doctor/session/{session-id}

Params

Name	Location	Type	Required	Description
session-id	path	string	yes	none

> Response Examples

> 200 Response

{"patient":"string","createdDate":0,"lastModifiedDate":0,"status":"CREATED","preferences":{"go

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	SessionDetailsDto

GET getPatientsOfDoctor

GET /doctor/patients

> Response Examples

> 200 Response

[{"username":"string","doctor":"string"}]

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	Inline

Responses Data Schema

HTTP Status Code **200**

Name	Type	Required	Restrictions	Title	description
anonymous	[Patient]	false	none		none
» username	string	false	none		none
» doctor	string	false	none		none

GET getPatientDetails

GET /doctor/patients/{patient-name}

Params

Name	Location	Type	Required	Description
patient-name	path	string	yes	none

> Response Examples 200 Response

{"username":"string","info":{"age":0,"sex":"MALE","lengthOfPsychosisInMonths":0,"known Diseases

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	PatientDetailsDto

GET getToken

GET /download/token

Params

Name	Location	Type	Required	Description
context	query	CoroutineContext	no	none

> Response Examples

> 200 Response

"string"

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	string

GET finishedSessions

GET /download/finished

Params

Name	Location	Type	Required	Description
accessToken	query	string	yes	none
context	query	CoroutineContext	no	none

> Response Examples

> 200 Response

"string"

Responses

HTTP Status Code	Meaning	Description	Data schema
200	OK	OK	string

4. Datenschema

CreateUserDto

{
"username": "string", "password": "string"
}

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
password	string	false	none		none

CreatePatientDto

{
"username": "string", "password": "string", "doctor": "string"
}

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
password	string	false	none		none
doctor	string	false	none		none

DrugPreference


```
{
"drug": "AMISULPRID",
"wouldTakeAgain": true
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
drug	string	false	none		none
wouldTakeAgain	boolean	false	none		none

Enum

Name	Value
drug	AMISULPRID
drug	ARIPIPRAZOL
drug	CARIPRAZIN
drug	CLOZAPIN
drug	HALOPERIDOL
drug	OLANZAPIN
drug	PALIPERIDON
drug	RISPERIDON
drug	PERPHENAZIN

drug	QUETIAPIN
Name	**Value**
drug	SERTINDOL
drug	ZIPRASIDON

Preferences


```
{
"goals": [ "string"
],
"drugsTakenBefore": [
{
"drug": "AMISULPRID",
"wouldTakeAgain": true
}
],
"otherDrugsTaken": [ "string"
],
"sideEffectsToAvoid": [ "EFFICACY"
],
"otherSideEffectsToAvoid": [ "string"
],
"thinkAboutMethod": true, "methodPreferences": [
"PILL"
],
"otherRemarks": [ "string"
]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
goals	[string]	false	none		none
drugsTakenBefore	[DrugPreference]	false	none		none
otherDrugsTaken	[string]	false	none		none
sideEffectsToAvoid	[string]	false	none		none

Name	Type	Required	Restrictions	Title	Description
otherSideEffectsToAvoid	[string]	false	none		none
thinkAboutMethod	boolean	false	none		none
methodPreferences	[string]	false	none		none
otherRemarks	[string]	false	none		none

MedicalInformation


```
{
"age": 0, "sex": "MALE",
"lengthOfPsychosisInMonths": 0, "knownDiseases": [
"CARDIAC"
],
"otherDiseases": [ "string"
],
"groups": [ "ADOLESCENT"
]
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
age	integer(int32)	false	none		none
sex	string	false	none		none
lengthOfPsychosisInMonths	integer(int32)	false	none		none
knownDiseases	[string]	false	none		none
otherDiseases	[string]	false	none		none
groups	[string]	false	none		none

Enum

Name	Value
sex	MALE
sex	FEMALE
sex	OTHER

ActionDto


```
{
"type": "string", "timestamp": 0, "payload": "string"
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
type	string	false	none		none
timestamp	integer(int64)	false	none		none
payload	string	false	none		none

Note


```
{
"commonReasons": [ "BAD_EFFICACY"
],
"otherReasons": "string"
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
commonReasons	[string]	false	none		none
Name	**Type**	**Required**	**Restrictions**	**Title**	**Description**

otherReasons	string	false	none		none

SessionLogDto


```
{
"actions": [
{
"type": "string", "timestamp": 0, "payload": "string"
}
],
"notes": {
"property1": { "commonReasons": [
"BAD_EFFICACY"
],
"otherReasons": "string"
},
"property2": { "commonReasons": [
"BAD_EFFICACY"
],
"otherReasons": "string"
}
}
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
actions	[ActionDto]	false	none		none
notes	object	false	none		none
» **additionalProperties**	Note	false	none		none

Decision


```
{
"selectedDrugs": [ "AMISULPRID"
```

],
"decisionRemarks": "string"
}

Attribute

Name	Type	Required	Restrictions	Title	Description
selectedDrugs	[string]	false	none		none
decisionRemarks	string	false	none		none

SdmaUserDto

{
"username": "string", "role": "ADMINISTRATOR"
}

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
role	string	false	none		none

Enum

Name	Value
role	ADMINISTRATOR
role	DOCTOR
role	PATIENT

Patient

{
"username": "string", "doctor": "string"

}

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
doctor	string	false	none		none

ContinuationObject

{
"context": {}
}

Attribute

Name	Type	Required	Restrictions	Title	Description
context	CoroutineContext	false	none		none

CoroutineContext

{}

Attribute

None

SessionDto

{
"sessionId": "string", "status": "string", "created": 0,
"lastModified": 0

}

Attribute

Name	Type	Required	Restrictions	Title	Description
sessionId	string	false	none		none
status	string	false	none		none
created	integer(int64)	false	none		none
lastModified	integer(int64)	false	none		none

SessionDetailsDto


```
{
"patient": "string", "createdDate": 0,
"lastModifiedDate": 0, "status": "CREATED", "preferences": {
"goals": [ "string"
],
"drugsTakenBefore": [
{
"drug": "AMISULPRID",
"wouldTakeAgain": true
}
],
"otherDrugsTaken": [ "string"
],
"sideEffectsToAvoid": [ "EFFICACY"
],
"otherSideEffectsToAvoid": [ "string"
],
"thinkAboutMethod": true, "methodPreferences": [
"PILL"
],
"otherRemarks": [ "string"
]
},
"medicalInformation": { "age": 0,
```

```
"sex": "MALE",
"lengthOfPsychosisInMonths": 0, "knownDiseases": [
"CARDIAC"
],
"otherDiseases": [ "string"
],
"groups": [ "ADOLESCENT"
]
},
"log": {
"actions": [
{
"type": "string", "timestamp": 0, "payload": "string"
}
],
"notes": {
"property1": { "commonReasons": [
"BAD_EFFICACY"
],
"otherReasons": "string"
},
"property2": { "commonReasons": [
"BAD_EFFICACY"
],
"otherReasons": "string"
}
}
},
"decision": { "selectedDrugs": [
"AMISULPRID"
],
"decisionRemarks": "string"
}
}
```


Attribute

Name	Type	Required	Restrictions	Title	Description
patient	string	false	none		none
createdDate	integer(int64)	false	none		none
lastModifiedDate	integer(int64)	false	none		none

Name	Type	Required	Restrictions	Title	Description
status	string	false	none		none
preferences	Preferences	false	none		none
medicalInformation	MedicalInformation	false	none		none
log	SessionLogDto	false	none		none
decision	Decision	false	none		none

Enum

Name	Value
status	CREATED
status	MEDICAL_INFO_PROVIDED
status	PREFERENCES_PROVIDED
status	ALL_INFO_PROVIDED
status	STARTED
status	FINISHED

PatientDetailsDto


```
{
"username": "string", "info": {
"age": 0,
"sex": "MALE",
"lengthOfPsychosisInMonths": 0, "knownDiseases": [
"CARDIAC"
],
"otherDiseases": [ "string"
],
"groups": [ "ADOLESCENT"
]
},
"sessions": [
{
```

"sessionId": "string", "status": "string",

"created": 0,
"lastModified": 0
}
]
}

Attribute

Name	Type	Required	Restrictions	Title	Description
username	string	false	none		none
info	MedicalInformation	false	none		none
sessions	[SessionDto]	false	none		none

5. Implementierung

5.1. Authentifizierung und Autorisierung

Absicherung von REST-Endpunkten

Alle Benutzer werden in der Datenbank mit ihrem Benutzernamen und dem bcrypt-Hash ihres Passworts gespeichert. Das Speichern nur des Hashes stellt sicher, dass Passwörter selbst im Falle eines Datenlecks nicht offengelegt werden. Grundsätzlich sollten alle Endpoints nur für authentifizierte Benutzer zugänglich sein.

Eine Möglichkeit der Authentifizierung wäre HTTP Basic Auth. Der Benutzer sendet seinen Benutzernamen und sein Passwort bei jeder REST-Anfrage. Das Passwort wird mit dem Salt aus dem gespeicherten bcrypt-Hash gehasht, und der resultierende Hash wird mit dem gespeicherten Hash verglichen, um den Benutzer zu authentifizieren.

Der Nachteil dieser Lösung ist, dass das Passwort im Klartext auf dem Client gespeichert werden müsste. Wenn der Benutzer vergisst, sich auszuloggen, oder das Passwort durch eine andere Website oder ein kompromittiertes Browserfenster offengelegt wird, hätte ein Angreifer uneingeschränkten Zugriff.

Dieses Problem kann umgangen werden, indem der Server bei der ersten Anfrage eine Session ID an den Client sendet. Bei allen folgenden Anfragen sendet der Client diese

Session ID, die den Benutzer identifiziert. Die Session ID ist nur für eine begrenzte Zeit gültig. Wird die Session ID kompromittiert, kann ein Angreifer sie nur bis zum Ablauf nutzen.

Der Nachteil dieser Lösung ist, dass der Server jede Session in der Datenbank oder im Speicher speichern muss, um die Session ID einem Benutzer zuzuordnen. Bei einer steigenden Anzahl gleichzeitiger Benutzeranfragen kann dies zu Performanceproblemen führen.

Um die SDMA-App skalierbar zu halten, werden JWT-Tokens verwendet. Ein neuer Login-Endpoint, der mit HTTP Basic Auth aufgerufen werden kann, stellt dem Client ein JWT-Token bereit. Dieses Token enthält alle Informationen über den Benutzer, einschließlich Name und Rolle, und wird mit dem geheimen Schlüssel des Servers verschlüsselt. Bei jeder weiteren REST-Anfrage sendet der Client das JWT-Token zur Authentifizierung. Der Server kann das Token mit seinem geheimen Schlüssel entschlüsseln, um die Benutzerinformationen zu erhalten. Die Benutzerinformationen können in den Spring Security Context injiziert werden, indem ein RequestFilter hinzugefügt wird..

```
@Component
class JwtRequestFilter(
    private val userDetailsService: UserDetailsService,
    private val jwtTokenUtil: JwtTokenUtil,
) : OncePerRequestFilter() {

    override fun doFilterInternal(request: HttpServletRequest,
                                  response: HttpServletResponse,
                                  filterChain: FilterChain) {
        if (SecurityContextHolder.getContext().authentication == null) {
            SecurityContextHolder.getContext().authentication =
                request.getHeader( name: "Authorization")
                    ?.takeIf { it.startsWith( prefix: "Bearer ") }
                    ?.substring( startIndex: 7)
                    ?.takeIf { jwtTokenUtil.validateToken(it) }
                    ?.let { userDetailsService.loadUserByUsername(jwtTokenUtil.extractUsername(it)) }
                    ?.let { UsernamePasswordAuthenticationToken(it, credentials: null, it.authorities) }
                    ?.also { it.details = WebAuthenticationDetailsSource().buildDetails(request) }
        }
        filterChain.doFilter(request, response)
    }
}
```

Darstellung 3: Das JWT-Token wird gelesen und die Benutzerinformationen werden in den Spring-Security-Kontext eingefügt

Nach der Authentifizierung muss die Autorisierung des Benutzers für die angefragte Methode überprüft werden. Die SDMA-App benötigt zwei Ebenen der Autorisierung:

1. Rollenbasierte Autorisierung: Einige Endpoints sind nur für Administratoren oder Ärzte zugänglich. Mit Spring Security kann auf Methodenebene festgelegt werden, welche Rollen Zugriff auf eine Methode haben, z. B. durch die Annotation @RolesAllowed.
2. Ressourcenbasierte Autorisierung: Jeder Benutzer darf nur auf die Daten zugreifen, auf die er Zugriff hat. Patienten können nur ihre eigenen Daten sehen, Ärzte nur die Daten ihrer zugewiesenen Patienten. Dies wird manuell überprüft, indem der Benutzer aus dem Security Context mit dem behandelnden Arzt oder dem Patienten der Session verglichen wird, wie in der Datenbank gespeichert.

Speichern des JWT-Tokens auf dem Client

Um auf geschützte REST-Endpoints zuzugreifen, ruft der Client einen Login-Endpoint über HTTP Basic Authentication auf. Dazu startet der Client mit einer Login-Seite, auf der der Benutzer nach Benutzernamen und Passwort gefragt wird. Nach Erhalt des JWT-Tokens ruft der Client einen Endpoint auf, um den Benutzernamen und die Rolle zu erhalten, die zu diesem JWT-Token gehören.

JWT-Token, Rolle und Benutzername werden anschließend in einem Redux Store gespeichert. Redux wird verwendet, um den Anwendungskontext zu speichern und von jeder React-Komponente darauf zuzugreifen.

Mit dem Redux Toolkit kann der Store einfach konfiguriert werden, indem für jeden unabhängigen Teil des Stores ein Slice erstellt wird. Für das User-Slice sind zwei Aktionen (Reducer) relevant:

- Logging in a new user: Speichern von Benutzername, Rolle und zugehörigem JWT-Token
- Logging out a current user: Löschen aller gespeicherten Benutzerinformationen aus dem Redux Store

Bei jeder weiteren REST-Anfrage liest der Client das JWT-Token aus dem Redux Store und verwendet es im Authorization-Header.

Die Konfiguration des Redux Stores und der Reducer erfolgt mit der createSlice()-

Methode des Redux Toolkit.

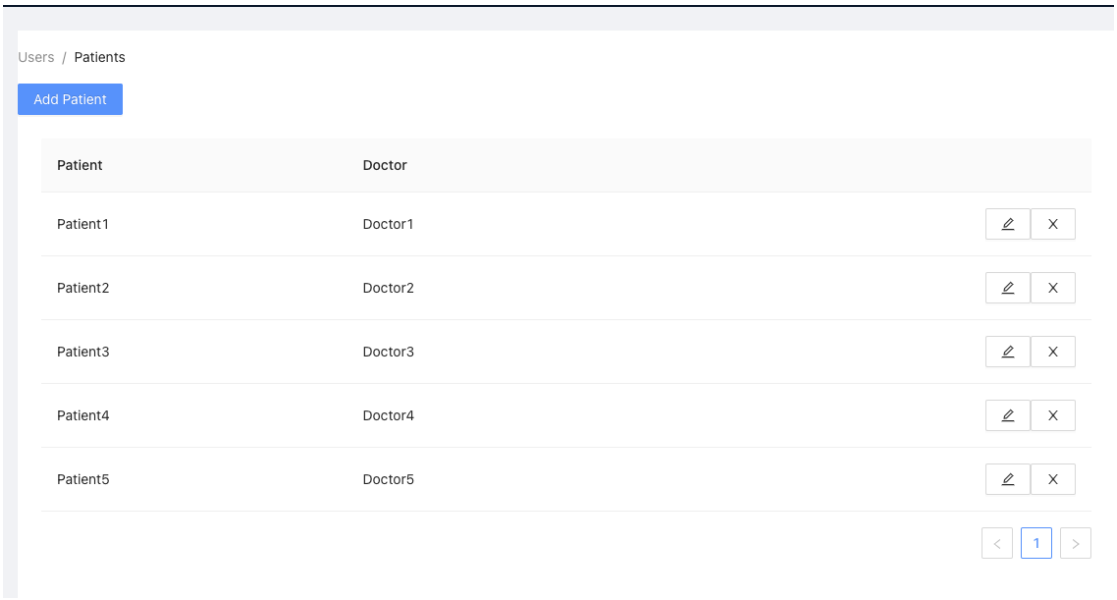
```
import { createSlice } from "@reduxjs/toolkit" export const ADMIN =
"ADMINISTRATOR"
export const DOCTOR = "DOCTOR" export const PATIENT = "PATIENT"

const nonUser = { username: "NONE", role: "None", jwtToken: "empty"
}

const initialState = { loggedIn: false, currentUser: nonUser
}

const userSlice = createSlice({ name: 'user',
initialState, reducers: {
loginUser(state, action) { state.currentUser = action.payload state.loggedIn = true
},
logout(state, action) { state.loggedIn = false state.currentUser = nonUser
}
}
})

export default userSlice.reducer

export const { loginUser, logout } = userSlice.actions

export const username = state => state.user.currentUser.username export const
isUserLoggedIn = state => state.user.loggedIn export const userRole = state =>
state.user.currentUser.role
export const currentJwtToken = state => state.user.currentUser.jwtToken
```

5.2. Benutzerpanels

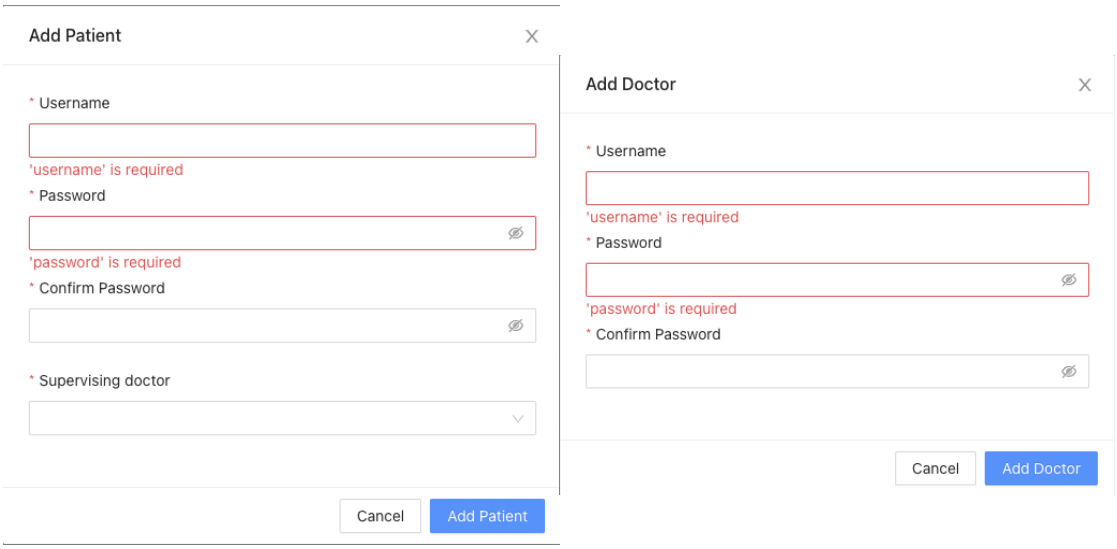
Wenn ein Administrator sich einloggt, zeigt die SDMA-Anwendung das Administration Panel an. Der Benutzer erhält eine Übersicht über bestehende Patienten und deren behandelnde Ärzte. Er kann neue Ärzte und Patienten erstellen sowie deren Pseudonyme und Passwörter festlegen. Beim Erstellen von Patienten kann der betreuende Arzt aus einer Liste aller registrierten Ärzte ausgewählt werden.

Der Administrator legt den Benutzernamen fest und ist damit verantwortlich für die Pseudonymisierung der Benutzerdaten. Der Direktor der medizinischen Studie sollte das

Administration Panel verwenden, da er alle teilnehmenden Ärzte und Patienten kennt und diese als SDMA-Benutzer anlegen kann. Je nach Anforderungen an die Pseudonymisierung der Patientendaten kann er die Benutzernamen selbst bestimmen.



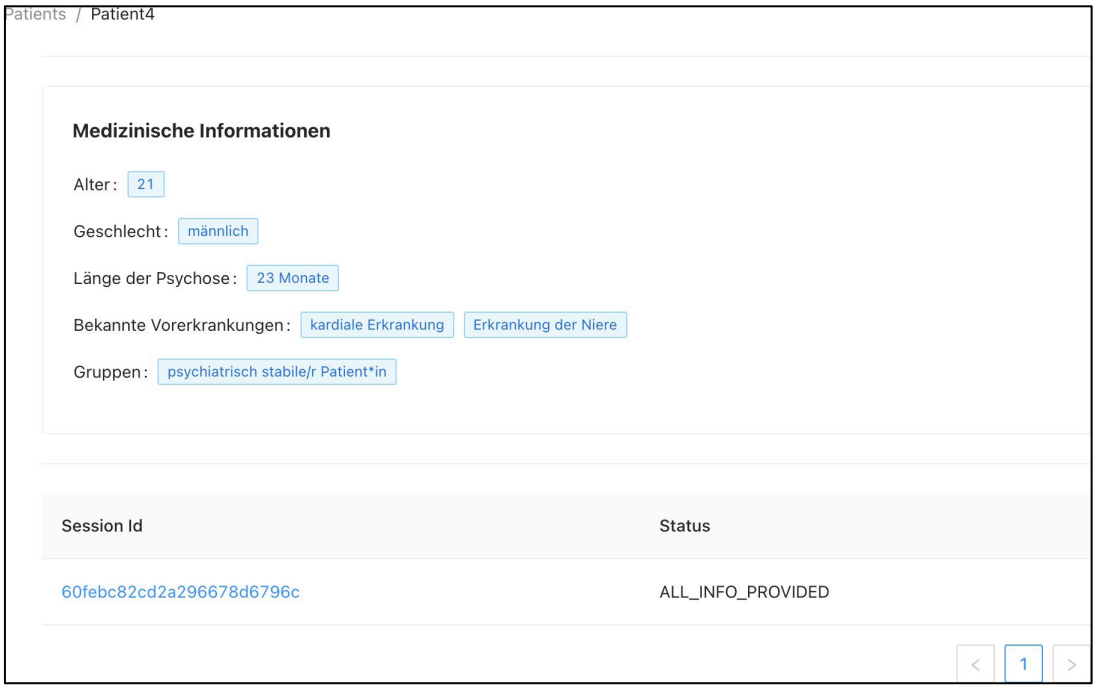
Darstellung 4: Liste der Patienten registriert im SDMA



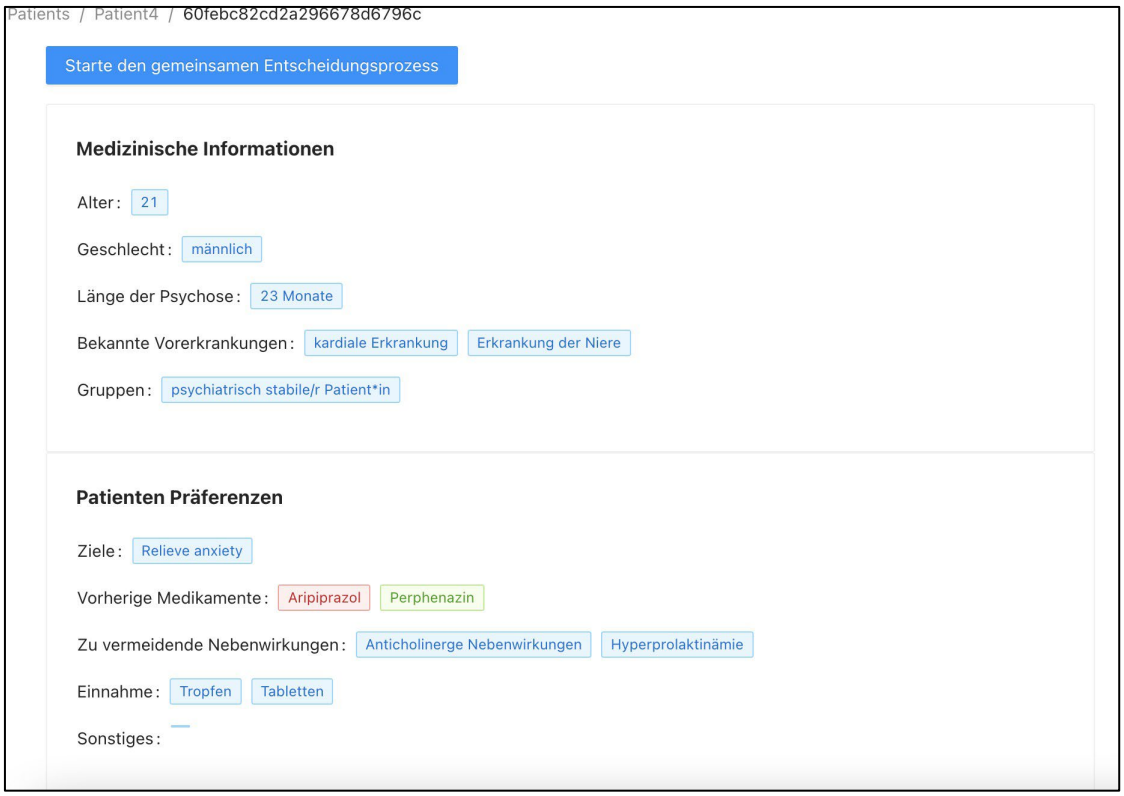
Darstellung 5: Erstellung eines Patienten und eines Behandlers

Ein Arzt sieht eine Liste aller Patienten, die ihm zugewiesen sind. Die verlinkte Detailansicht eines Patienten enthält alle erstellten Sessions sowie die medizinischen Informationen des Patienten, sofern der Arzt diese bereits bereitgestellt hat.

Der Arzt kann eine neue Session auf der Patientendetailseite erstellen, sofern alle vorherigen Sessions abgeschlossen sind. Die Session-Detailansicht erreicht er durch Anklicken einer Session in der Liste. Die Session-Detailseite enthält alle während der Session bereitgestellten Informationen, einschließlich medizinischer Daten, Präferenzen und der endgültigen Entscheidung. Abhängig vom Zustand der Session kann der Arzt wählen, ob er zuerst die medizinischen Informationen des Patienten bereitstellt oder direkt den Shared-Decision-Making-Prozess startet.



Darstellung 6: The patient details page



Darstellung 7: Die Seite über die Sitzungsdetails

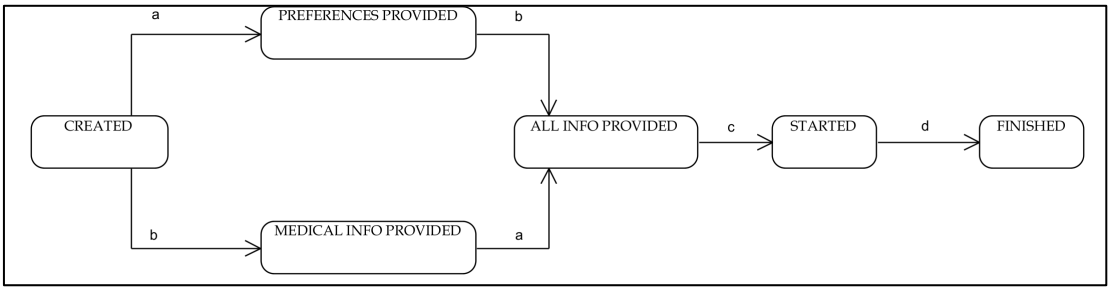
5.3. *Sitzungen (Session)*

Eine Session spiegelt den Prozess der Medikamentenauswahl innerhalb des Entscheidungsassistenten wider. Der Arzt erstellt die Session für einen bestimmten Patienten. Nach der Erstellung kann der Arzt die medizinischen Informationen des Patienten einreichen, und der Patient kann seine Präferenzen zur Medikation übermitteln. Diese beiden Schritte sind unabhängig voneinander und können in beliebiger Reihenfolge erfolgen. Nachdem sowohl die medizinischen Informationen als auch die Patientenpräferenzen vorliegen, kann der Arzt die Session starten und gemeinsam mit dem Patienten über die Medikation entscheiden. Nach dem Speichern der Entscheidung ist die Session abgeschlossen und kann nicht mehr verändert werden.

Eine Session kann mehrere Status haben:

1. CREATED: Die Session wurde von einem Arzt für einen bestimmten Patienten erstellt; es können keine weiteren Sessions für diesen Patienten erstellt werden.
2. PREFERENCES PROVIDED: Der Patient hat seine Präferenzen bezüglich Medikamente und Nebenwirkungen übermittelt.

3. MEDICAL INFO PROVIDED: Der Arzt hat die relevanten medizinischen Informationen des Patienten bereitgestellt.
4. ALL INFO PROVIDED: Sowohl medizinische Informationen als auch Patientenpräferenzen liegen vor; die Session ist bereit für die gemeinsame Entscheidungsfindung.
5. STARTED: Der Shared-Decision-Prozess wurde bereits gestartet; ein vorläufiger Fortschritt wurde gespeichert.
6. FINISHED: Patient und Arzt haben eine Entscheidung getroffen und ein Medikament ausgewählt. Zu diesem Zeitpunkt können keine Informationen mehr bearbeitet werden, und es kann eine neue Session erstellt werden.



Darstellung 8: Das Zustandsdiagram einer Sitzung

a) Der Patient äußert seine Behandlungswünsche.
b) Der Arzt stellt die medizinischen Daten des Patienten bereit.
c) Arzt und Patient besprechen die Behandlung gemeinsam mithilfe des SDMA.
d) Arzt und Patient entscheiden sich für ein Medikament und speichern die Entscheidung im SDMA.
Der Server stellt diesen Sitzungsablauf mithilfe eines Zustandsautomaten sicher. Bevor Änderungen an der Sitzung vorgenommen werden, wird das entsprechende Ereignis in der betreffenden Sitzung registriert. Handelt es sich bei dem Ereignis nicht um einen gültigen Übergang vom aktuellen Sitzungszustand, verwirft der Zustandsautomat das Ereignis und löst eine Ausnahme aus. Gehört das Ereignis zu einem gültigen Übergang, wird der resultierende Sitzungszustand zurückgegeben und in der Sitzung gespeichert..

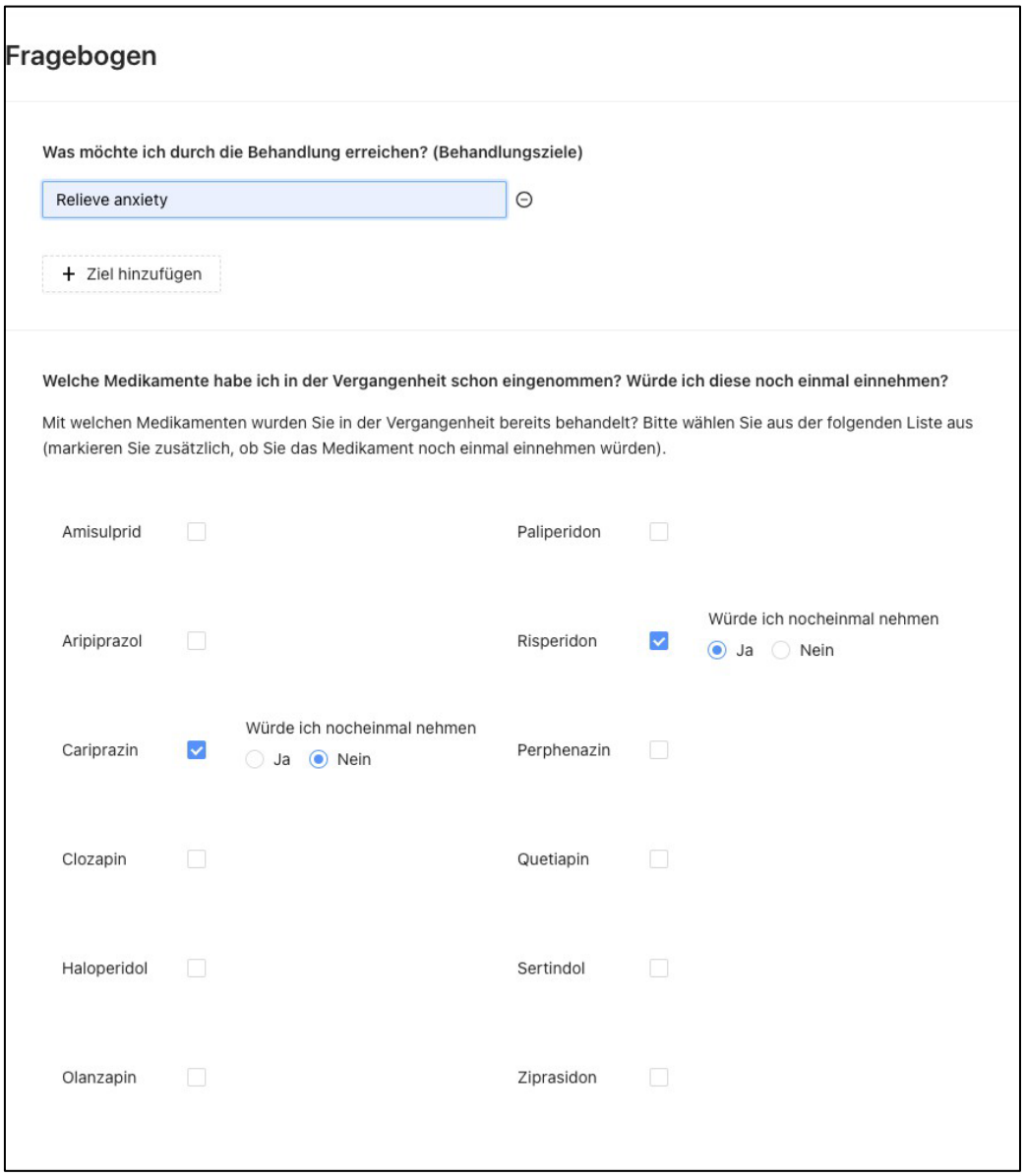
3.4 Preferences and Medical Information forms

Patient form

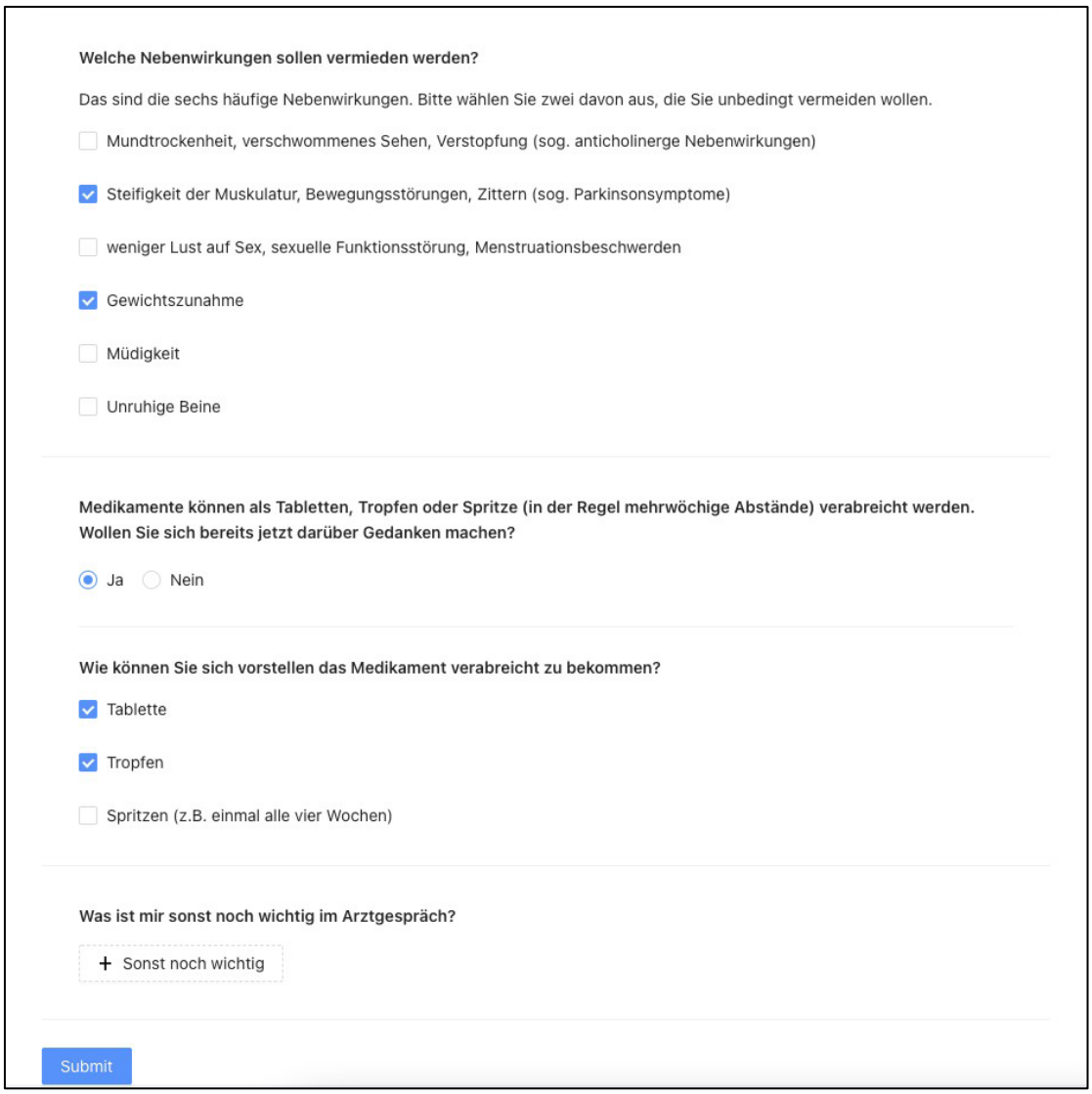
Die Formulare für die Patientenpräferenzen und die medizinischen Informationen werden mit Ant Design Form-Komponenten erstellt. Jede Frage hat ihr eigenes Format, abhängig von den erwarteten Antworttypen. Ziel ist es, das Beantworten der Fragen so einfach wie möglich zu gestalten, um Patienten mit möglicherweise eingeschränkter mentaler Leistungsfähigkeit die Teilnahme zu ermöglichen und Ärzten Zeit zu sparen,

wenn sie die medizinischen Informationen ausfüllen.

- Alter und Dauer der Psychose: Zahleneingabefelder, die nur numerische Eingaben erlauben.
- Geschlecht des Patienten: Radio-Button-Auswahl, da nur eine Antwort sinnvoll ist.
- Aktuelle Medikation, Begleiterkrankungen und Subgruppen: Checkbox-Gruppen, bei denen der Arzt alle zutreffenden Optionen für den Patienten auswählen kann.



Darstellung 9: Formular zur Erfassung der Patientenpräferenzen (Teil 1)



Darstellung 10: Formular zur Erfassung der Patientenpräferenzen (Teil 2)

Behandlerformular

Bei der Abfrage der Ziele und zusätzlichen Wünsche des Patienten kann der Patient freien Text eingeben und diesen in mehrere unabhängige Punkte strukturieren. Eine Textzeile steht standardmäßig zur Verfügung, weitere Zeilen können hinzugefügt oder entfernt werden, um zusätzliche Ziele oder Wünsche einzutragen. Patientenpräferenzen zu Antipsychotika: Checkbox-Gruppe ähnlich wie bei den Arztformularen. Für jedes ausgewählte Antipsychotikum kann der Patient angeben, ob er es erneut nehmen würde (Radio-Button mit "Ja" / "Nein"). Nebenwirkungen: Eine weitere Checkbox-Gruppe listet die Nebenwirkungen auf, die der Patient vermeiden möchte. Ursprünglich wurden alle

Antworten beim Absenden validiert, z. B. durfte ein Patient nur maximal 4 Nebenwirkungen auswählen, um Priorisierung zu erzwingen. Patiententests während der Entwicklung zeigten jedoch, dass solche Einschränkungen Patienten eher verwirren. Daher wurden die Regeln entfernt, und das Datenmodell sowie der Code wurden so angepasst, dass Fragen unbeantwortet bleiben können.

Darstellung 11: Das Formular für die medizinischen Informationen über den Patienten

Informationsspeicherung

Sobald der Patient oder der Arzt das Formular absendet, ruft der Client den entsprechenden HTTP-Endpunkt mit den Antworten im JSON-Body auf. Der Server verarbeitet die HTTP-Anfrage im Controller und prüft die Autorisierung wie oben beschrieben. Anschließend ruft der Controller die Serviceschicht auf, in der die Zustandsmaschine das entsprechende Ereignis registriert und der Sitzungsstatus wie im Abschnitt „Sitzung“ beschrieben aktualisiert wird. Schließlich speichert das Repository die medizinischen Informationen oder die Patienteneinstellungen in der Sitzung, sofern die Zustandsmaschine keine Ausnahme auslöst.

3.5 Gemeinsame Entscheidungsfindung (Shared Decision Making)

Der zentrale Bestandteil des Shared Decision Assistants ist die Darstellung der wissenschaftlichen Daten zu Antipsychotika für Patient und Arzt. Die Anwendung zeigt den mittleren relativen Risiko-Wert sowie die obere und untere Grenze für jedes Medikament und jede Nebenwirkung an.


```
{
    "ci_lb": "0.84",
    "ci_up": "2.2",
    "mean": "1.42",
    "quantile": "1",
    "rank": "14",
    "type": "adverse event"
}
```


Darstellung 12: The JSON representation of one anti-psychotic/side-effect combination

Erstellung der Datendiagramme

Die Graphen werden mit der JavaScript Library chart.js erstellt. Im Gegensatz zu anderen Bibliotheken wie ApexCharts oder FusionCharts kann hier die Darstellung der Daten individuell erzeugt werden.

Scatter Chart mit zwei Datasets:

Dataset: Konfidenzintervall (untere und obere Grenze) – Linie zwischen den Punkten,

Punkte selbst unsichtbar (radius = 0).

Dataset: Mittleres relatives Risiko – zeigt einen Punkt auf der Linie des Konfidenzintervalls.

Die Daten stammen aus einem JSON-Objekt (siehe Darstellung 12) für jede Kombination aus Antipsychotikum und Nebenwirkung, bereitgestellt vom Team von Prof. Leucht [5].

- Quantile: Beschreibt, zu welchem Drittel das Medikament bezüglich des relativen Risikos gehört.
- Farbgebung: Anfangs wurde eine Ampelfarbskala verwendet (rot = schlechteste, grün = beste), aber Experten empfahlen eine kontinuierliche Farbskala. Mit dem HSL-Farbsystem wird die Farbe auf Basis des mittleren relativen Risikos im Verhältnis zum maximalen Risiko aller Medikamente berechnet.

```
new Chart(ctx, {
    type: 'scatter',
    data: {
        datasets: [
            {
                data: [
                    {
                        x: data.ci_lb,
                        y: 0
                    },
                    {
                        x: data.ci_up,
                        y: 0
                    }
```

Darstellung 13: Die (teilweise) Konfiguration des chart.js-Diagramms

Das in Darstellung 12 dargestellte Quantil beschreibt, in welchem 33,3%-Quantil das Medikament liegt. Ein Antipsychotikum mit Quantil 3 befindet sich unter den 33,3 % der Antipsychotika mit dem höchsten relativen Risiko für diese Nebenwirkung. In einer ersten Version der Anwendung wurden die Grafiken je nach ihrem 33%-Quantil eingefärbt. Das höchste Quantil wurde rot, das mittlere gelb und das niedrigste grün für Nebenwirkungen dargestellt, wobei die Farbgebung für Wirksamkeit umgekehrt war. Die Farbgestaltung sollte Patienten helfen, das Diagramm leichter zu verstehen, da Rot intuitiv für „schlechter“ und Grün für „besser“ steht. Diese Theorie wurde in späteren

Tests mit Patienten bestätigt, die die Farbgebung als sehr intuitiv und hilfreich empfanden. Während Diskussionen mit Experten für „Shared Decision Aids“ stellten die medizinischen Stakeholder jedoch fest, dass die Ampelfarbgebung zu suggestiv sei. Besonders da zwei Antipsychotika unterschiedlicher Quantile oft im tatsächlichen relativen Risiko oder in der relativen Wirksamkeit sehr nahe beieinanderliegen können. Eine so expressive Farbgestaltung könnte hier die tatsächlichen Unterschiede zwischen zwei Antipsychotika überbewerten. Stattdessen schlugen die Stakeholder eine kontinuierliche Farbskala vor. Mithilfe des HSL-Farbsystems wird die Farbe der Grafik basierend auf dem mittleren relativen Risiko des Medikaments und dem maximalen relativen Risiko der spezifischen Nebenwirkung aller Medikamente generiert. Während der Farbton (Hue) auf 240 und die Sättigung (Saturation) auf 100 für eine blaue Farbskala gesetzt werden, wird die Helligkeit (Lightness) dynamisch berechnet.

$$Lightness = 50 + \frac{mean}{max} * 50$$

Anordnung der Diagramme

Ziel des Entscheidungsassistenten ist es, einen Vergleich zwischen den Antipsychotika zu ermöglichen. Außerdem muss der Patient das relative Risiko verschiedener Nebenwirkungen und die relative Wirksamkeit gegeneinander abwägen. Deshalb werden die Diagramme zu relativen Risiken und Wirksamkeit in einer Tabelle dargestellt. Die Zeilen bestehen aus den Antipsychotika, die Spalten aus den Nebenwirkungen und der Wirksamkeit.

Die Anzeige aller Nebenwirkungen in dieser Tabelle wäre für die meisten Patienten sehr überwältigend und würde auf die meisten Bildschirmgrößen nicht passen. Daher zeigt die App im Initialzustand nur die relative Wirksamkeit, da diese Information für Patienten unabhängig von ihren Präferenzen besonders wichtig ist. Der Patient und der Arzt können anschließend weitere Nebenwirkungen zu dem Vergleich hinzufügen, an denen sie besonderes Interesse haben. Dies könnten die Nebenwirkungen sein, die der Patient am meisten vermeiden möchte. Der Patient und der Arzt können außerdem Antipsychotika, die als ungeeignet gelten, aus der Tabelle entfernen, um die Auswahlmöglichkeiten einzugrenzen.

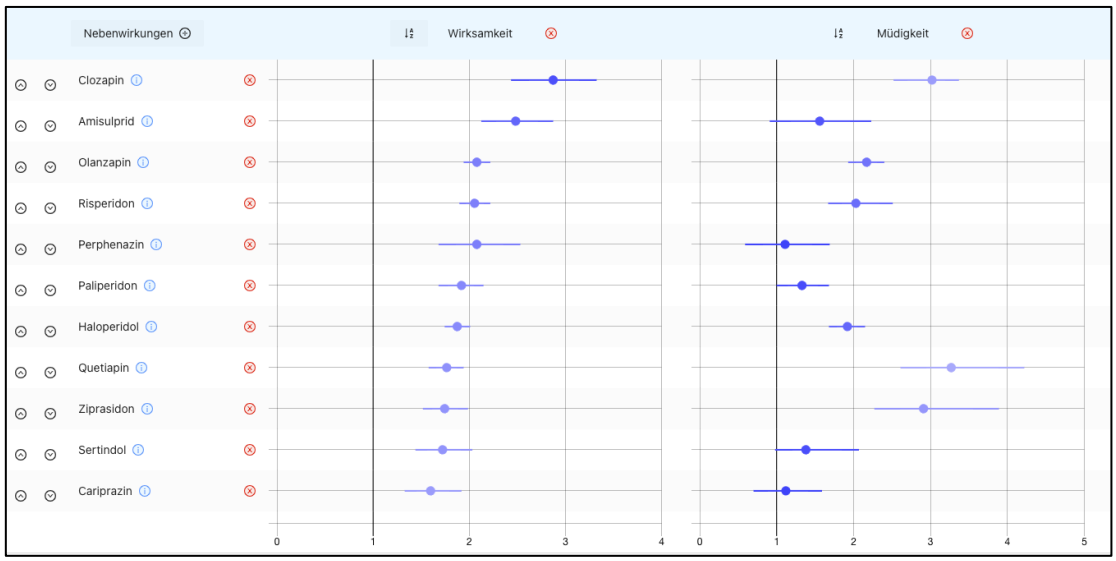
Redux verwaltet den internen Zustand der Tabelle. Analog zum Vorgehen bei den

Benutzeranmeldedaten erstellt das Redux Toolkit einen Redux-Slice sowohl für die Antipsychotika als auch für die Nebenwirkungen (einschließlich der Wirksamkeit). Der Redux-Zustand für beide besteht aus einer Liste aller angezeigten Nebenwirkungen/Antipsychotika sowie einer Liste der ausgeschlossenen..

```
{
    drugs: {
        included: ["AMISULPRID", "ARIPIPRAZOL", "CARIPRAZIN", "CLOZAPIN",
            "HALOPERIDOL", "OLANZAPIN", "PALIPERIDON", "RISPERIDON",
            "PERPHENAZIN", "QUETIAPIN", "SERTINDOL", "ZIPRASIDON"],
        excluded: []
    },
    effects: {
        included: ["EFFICACY", "FATIGUE"],
        excluded: ["ANTICHOLINERGIC", "WEIGHT_GAIN", "PARKINSON",
            "HYPERPROLACTINEMIA", "RESTLESS"]
    }
}
```

Darstellung 14: Redux-Zustand, der den aktuellen Zustand der Tabelle repräsentiert

Die Speicherung der ausgeschlossenen Elemente im Zustand erleichtert deren späteres Hinzufügen zu den angezeigten Elementen. Immer wenn die React-Komponente die Tabelle rendert, werden alle enthaltenen Antipsychotika und Nebenwirkungen abgerufen und für jede Kombination ein Diagramm erstellt. Ändert sich der Redux-Zustand aufgrund von Benutzereingaben, rendert die React-Komponente die Tabelle automatisch neu und erstellt oder entfernt die erforderlichen Diagramme, ohne alle unveränderten Diagramme neu rendern zu müssen.



Darstellung 15: Vergleichsansicht hinsichtlich Wirksamkeit und Ermüdung

Um einen guten Vergleich aller relativen Risiken für eine Nebenwirkung zu gewährleisten, müssen die Diagramme in einer Spalte alle die gleiche Skala haben. Andernfalls könnte für eine Nebenwirkung der Punkt im Diagramm weiter rechts liegen als bei einer anderen Nebenwirkung, obwohl das relative Risiko niedriger ist, weil die Skala größer ist. Eine feste Skala ist jedoch keine Option, da die Werte je nach Nebenwirkung stark variieren können.

Selbst wenn die Antipsychotika mit extremen relativen Risiken entfernt werden, können Antipsychotika mit ähnlichen relativen Risiken schwer zu vergleichen sein, da die Skala so groß ist. Eine dynamische Anpassung der Skala basierend auf allen Antipsychotika, die noch im Vergleich sind, hilft Patient und Arzt, Unterschiede jederzeit zu erkennen.

Die Skalen werden ebenfalls innerhalb eines Redux-Slices verwaltet. Sie hängen sowohl vom Zustand der Antipsychotika als auch vom Zustand der Nebenwirkungen ab. Um die Skalen zu berechnen, müssen die eingeschlossenen Antipsychotika bekannt sein, da die Skala von deren Werten abhängt, und nur die Skalen der eingeschlossenen Nebenwirkungen sollten berechnet werden. Daher berechnet der im zuständigen Redux-Slice definierte Reducer die Skalen jedes Mal neu, wenn sich einer der Zustände ändert. Der Redux-Zustand besteht aus einer Skala für jede aktuell eingeschlossene Nebenwirkung. Ein Skalenobjekt besteht aus einem Maximum für das Diagramm und einer Schrittweite. Das Maximum ist die Decke des höchsten oberen Konfidenzintervalls aller eingeschlossenen Antipsychotika. Die Schrittweite wird als Maximum geteilt durch

fünf definiert, um das Diagramm nicht mit Linien für größere Werte zu überladen.
Beim Rendern der Tabelle werden alle notwendigen Skalen einmal aus dem Redux-Zustand abgerufen und dann an die Graph-Komponenten weitergegeben, um die Skala des Diagramms zu konfigurieren. Zusätzlich wird am unteren Ende der Tabelle eine weitere Zeile von Diagrammen gerendert. Diese Zeile enthält keine Daten, zeigt aber die Skala für alle darüberliegenden Zeilen an, wobei die Skala selbst nicht angezeigt wird. Dies macht die Übersicht über mehrere übereinander gestapelte Diagramme deutlich weniger verwirrend und leichter verständlich, da sie wie ein einzelnes Diagramm mit mehreren Datenpunkten wirken.

Interaktion mit den Datendiagrammen ermöglichen

Die Patienten und Ärzte sollen ebenfalls mit dem Shared Decision-Making-Assistant (SDMA) interagieren können. Wie bereits erwähnt, können sie zunächst weitere Nebenwirkungen zum Vergleich hinzufügen, da im Initialzustand nur die relative Wirksamkeit angezeigt wird. In der oberen linken Ecke der Tabelle wird ein Button dargestellt, der ein Dropdown-Menü öffnet. Für jede Nebenwirkung in der „nicht eingeschlossen“-Liste des Redux-Zustands wird ein Menüeintrag mit der Menu-Komponente von Ant Design erstellt. Durch die Auswahl einer Nebenwirkung wird eine im Redux-Store registrierte Aktion ausgelöst, die diese Nebenwirkung aus der „nicht eingeschlossen“-Liste entfernt und der „eingeschlossen“-Liste hinzufügt. Wie oben beschrieben, rendert sich die Tabelle automatisch neu und fügt eine weitere Spalte mit den Daten der ausgewählten Nebenwirkung hinzu.
Wenn die Nutzer Nebenwirkungen zum Vergleich hinzufügen, müssen sie irgendwann auch Nebenwirkungen aus der Übersicht entfernen, an denen sie kein Interesse mehr haben. Im Spaltenkopf neben dem Namen der Nebenwirkung wird ein roter Entfernen-Button angezeigt, der eine weitere Aktion auslöst. Diese Aktion verschiebt die Nebenwirkung im Redux-Zustand von der „eingeschlossen“-Liste zurück in die „nicht eingeschlossen“-Liste.
Parallel zum Ein- und Ausschließen von Nebenwirkungen bietet der SDMA den Nutzern die gleichen Möglichkeiten für Antipsychotika. Im Zeilenkopf befindet sich ein Entfernen-Button. Dieser öffnet ein Modal-Fenster, in dem der Patient oder Arzt angeben kann, warum dieses Antipsychotikum ausgeschlossen wurde. Diese Freitexteingabe wird

gespeichert und kann später verwendet werden, um die Entscheidung nachzuvollziehen und zu bewerten, ob die Begründung noch gültig ist. Nach Bestätigung des Ausschlusses wird – wie bei den Nebenwirkungen – eine Aktion ausgelöst, und das Antipsychotikum wird im Redux-Zustand verschoben. Wie alle anderen UI-Komponenten stammen das Textfeld und das Modal-Fenster aus der Ant Design-Bibliothek.
Um alle ausgeschlossenen Antipsychotika einsehen zu können, befindet sich unter der Tabelle mit den Graphen eine aufklappbare Tabelle. Diese zeigt den Namen des Antipsychotikums, die angegebene Begründung des Patienten für den Ausschluss sowie einen Button, um das Antipsychotikum wieder in den Vergleich einzufügen.

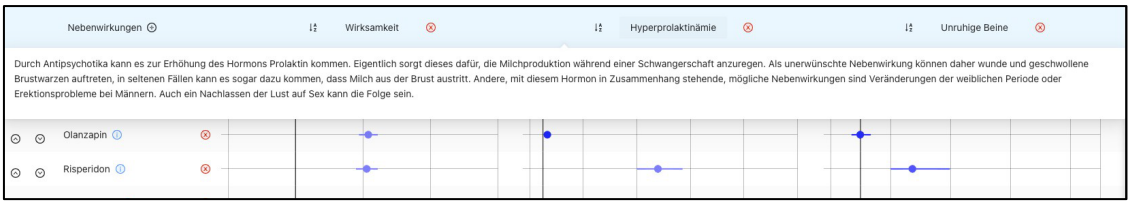
Darstellung 16: Der Dialog zur Ausschluss eines Antipsychotikums aus dem Vergleich

Im Spaltenkopf kann der Patient auf den Sortier-Button klicken, um die Antipsychotika in der Tabelle für diese spezielle Nebenwirkung oder Wirksamkeit von „am besten“ bis „am schlechtesten“ zu sortieren. Die Reihenfolge der Antipsychotika hängt dabei nicht nur vom Mittelwert des relativen Risikos bzw. der Wirksamkeit ab, sondern auch davon, wie breit das Konfidenzintervall ist. Dieses Ranking, wie in Abbildung 12 dargestellt, wird vom medizinischen Team bereitgestellt, das auch die übrigen Daten liefert.
Diese Funktion soll einen leicht verständlichen Überblick über alle Antipsychotika geben, wenn der Fokus auf nur einer Nebenwirkung liegt. Die Sortierung erfolgt wieder über Manipulation des Redux-Zustands innerhalb eines Reducers. Der Reducer ruft die Daten (siehe Abbildung 12) aller eingeschlossenen Antipsychotika für die zu sortierende Nebenwirkung oder Wirksamkeit ab und sortiert dann die „eingeschlossen“-Liste anhand des Rank-Attributs. Da beim Rendern der Tabelle über Redux iteriert wird, verändert eine Änderung der Reihenfolge der Elemente in der Liste die Reihenfolge der Zeilen in der Tabelle, ohne dass die Graphen neu gerendert werden müssen. Das DOM ordnet

die bereits gerenderten Graph-Komponenten automatisch neu an.
Ähnlich erlaubt die interaktive Übersicht dem Patienten, Zeilen nach oben oder unten zu verschieben. So kann der Patient zwei Antipsychotika nebeneinanderstellen, um sie effizienter zu vergleichen, oder das aktuelle Favoriten-Antipsychotikum nach oben setzen. Diese Aktionen manipulieren den Redux-Zustand, indem das Antipsychotikum mit dem Element davor oder dahinter getauscht wird, wenn es nach oben oder unten verschoben wird.

Anpassungen während der Entwicklung

Da die interaktive Graph-Übersicht der zentrale Teil des Shared Decision-Making-Assistenten ist, wurde sie von Anfang an als Prototyp entwickelt und von diesem Zeitpunkt an mit tatsächlichen psychotischen Patienten in der MRT-Klinik getestet. Die initialen Anforderungen und Implementierungen wurden während dieser Tests und in regelmäßigen Meetings mit den medizinischen Stakeholdern in der MRT verfeinert.
Mehrere Patienten hatten während der Tests Schwierigkeiten, die Graphen den Antipsychotika zuzuordnen. Um das Verfolgen der Zeilen zu erleichtern, entschieden sich die Stakeholder dafür, die Spalten in abwechselnden Farben einzufärben. Mittels CSS wurde jede zweite Tabellenzelle hellblau eingefärbt. In späteren Tests beschwerte sich kein Patient mehr darüber, die Zeilen nicht mehr verfolgen zu können, sodass dieses Problem scheinbar gelöst war.
Patienten hatten außerdem Schwierigkeiten, die Nebenwirkungen zu verstehen. Medizinische Begriffe wie Hyperprolaktinämie wurden nicht verstanden. Um dem entgegenzuwirken, erscheint beim Anklicken einer Nebenwirkung eine kurze Erklärung der jeweiligen Nebenwirkung..



Darstellung 17: Kurze Erklärung der Nebenwirkungen

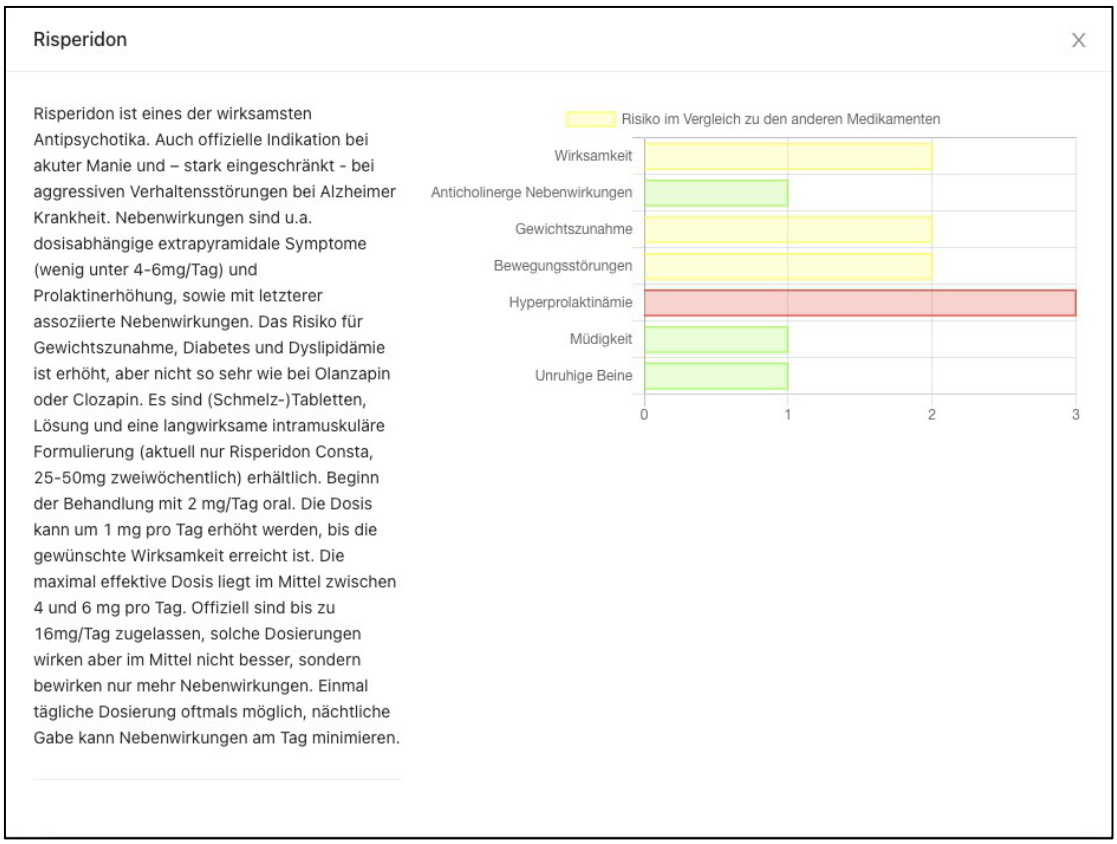
Zusätzlich vermissten Patienten die Möglichkeit, eine Übersicht über ein bestimmtes Antipsychotikum zu erhalten. Viele Patienten wollten alle Nebenwirkungen des

Medikaments sehen, das sie aktuell einnahmen oder in der Vergangenheit eingenommen hatten. Im Prototyp mussten die Nutzer jede Nebenwirkung einzeln zum Vergleich hinzufügen, wodurch die Übersicht sehr unübersichtlich wurde.

Darüber hinaus wollte das medizinische Team einen kurzen Informationstext über das Antipsychotikum für den Arzt einfügen. Dieser Informationstext enthält eine kurze Beschreibung, Hinweise, wann das Antipsychotikum nicht verschrieben werden sollte, sowie Angaben zur Dosierung. Der Text soll Ärzte an kritische Fakten zu einem Antipsychotikum erinnern, mit dem sie möglicherweise schon Erfahrung haben

Um beide neuen Anforderungen zu erfüllen, enthält die Übersicht ein Informations-Modal für jedes Antipsychotikum. Dieses Modal öffnet sich beim Anklicken des Antipsychotikums in der Tabelle, erkennbar durch ein Informationssymbol am Ende jeder Zeile. Auf der linken Seite wird die Zusammenfassung des Antipsychotikums angezeigt. Auf der rechten Seite befindet sich ein Diagramm, das zeigt, zu welchem 33%-Quantil das Antipsychotikum für alle Nebenwirkungen und die Wirksamkeit gehört.

Zur Darstellung des Quantils wird ein vertikales Balkendiagramm mit Chart.js verwendet, bei dem jeder Balken eine Nebenwirkung repräsentiert. Jeder Balken zeigt den Quantilwert, wie in Abbildung 12 dargestellt, und ist in dem Rot-Gelb-Grün-Schema eingefärbt, das ursprünglich auch für die Diagramme in der Übersichtstabelle vorgesehen war. Alle zuvor diskutierten Probleme mit dieser Farbgebung bleiben bestehen. Da jedoch keine bessere Darstellungsmöglichkeit zur Klassifizierung der relativen Risiken von Antipsychotika verfügbar ist, entschieden die Stakeholder, dieses Diagramm beizubehalten, um Patienten und Ärzten eine bessere Übersicht zu ermöglichen.



Darstellung 18: Das Informationsmodal für Antipsychotika

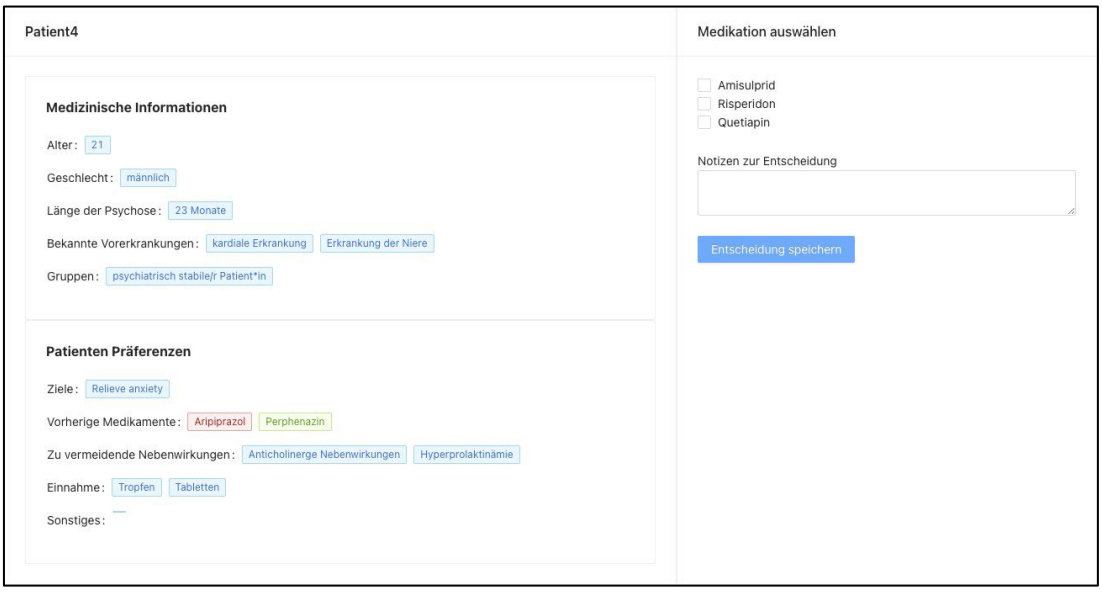
Finale Entscheidung

Die interaktiven Daten-Graphen werden in der Entscheidungsphase einer Sitzung platziert. Der Nutzer kann frei zwischen den interaktiven Diagrammen und der Zusammenfassung aller zuvor in den Formularen gesammelten Daten wechseln, wie in den vorherigen Abschnitten beschrieben. Der Client ruft die Sitzung vom Backend-Server ab, einschließlich der medizinischen Informationen und der Präferenzen des Patienten. Diese Informationen werden dann mithilfe der Komponenten von Ant Design zur Datenanzeige dargestellt.

Ziel ist es, dass Patient und Arzt sich vorherige Antworten – wie die Präferenzen des Patienten oder medizinische Vorerkrankungen – ansehen und ihre Entscheidungen auf dieser Basis treffen, während sie die interaktiven Daten-Graphen nutzen. Die Patientenübersicht und die interaktive Oberfläche befinden sich auf derselben reaktiven Seite, sodass das Hin- und Herwechseln zwischen den Ansichten nahtlos möglich ist.

Nachdem Patient und Arzt die interaktiven Daten-Graphen bearbeitet haben, können sie zur Auswahlseite weitergehen. Neben der Übersicht über die Patienteninformationen

befindet sich eine Checkbox-Gruppe mit allen Antipsychotika, die im vorherigen Schritt nicht ausgeschlossen wurden. Die Nutzer können eine Kombination von Antipsychotika auswählen und einen Kommentar zu ihrer Entscheidung hinzufügen.
Die interaktiven Daten-Graphen dienen dazu, dass Patient und Arzt Antipsychotika basierend auf ihrem Nebenwirkungsprofil einzeln ausschließen. Nach dem Ausschluss einiger Antipsychotika haben die Nutzer jedoch möglicherweise bereits ein klares Bild davon, welches Medikament sie bevorzugen. Die Anwendung zwingt sie nicht, alle anderen Antipsychotika aus der Vergleichstabelle auszuschließen. Sie können die Antipsychotika markieren, die sie für ihre Medikation bevorzugen, oft bestehend aus einem oder zwei Antipsychotika. Patient und Arzt können anschließend kommentieren, warum sie diese Wahl getroffen haben und welche Dosierung sie festgelegt haben.
Der SDMA-Client speichert die Entscheidungsdaten, indem er beim Absenden der Entscheidung eine HTTP-Anfrage an den SDMA-Server sendet..



Darstellung 19: Die Seite, auf der Patient und Arzt die endgültige Entscheidung treffen.