Web App - for study management

Geck, Nicole Gkazgkalidou, Eleni Heitlinger, Marius
Krüger, Julika Kulla, Lucas Leibensperger, David

January 11, 2023

Heilbronn University of Applied Sciences
Ruprecht Karl University of Heidelberg
Medizinische Informatik Bachelor - Semester 5
Anwendungsbezogene Medizinische Informatik
Lecturer: Max Wolfgang Seitz

Abstract

Period and cycle tracking are important tools for managing reproductive health and family planning. They can help individuals understand their menstrual cycle and make informed decisions about contraception and pregnancy. There are various helpful methods that support individuals in charting their cycles. One of these methods is Sensiplan©. Sensiplan© is a paper-based methodology that can help determine fertility windows as well as visualize and interpret menstrual cycles. To compare Sensiplans© overall effectiveness against digital period trackers, a research database was set up, in which study nurses manually manage data from both Sensiplan© and period trackers. Considering this is both labor intensive and prone to errors, a web application supporting study nurses and researchers in maintaining the database was conceptualised. To realize this web application, a back-, and front-end were developed. The back-end implements the Fast Healthcare Interoperability Resources' (FHIR) standard, which was chosen due to its flexibility and interoperability. Participant related data is stored in the FHIR resources *patient* and *questionnaireResponse*. The back-end acts as middle-ware between a public FHIR server, used for testing and validation purposes, and the front-end. Communication is realized via a REST interface using Hypertext Transfer Protocol (HTTP) methods covering create, read, update and delete (CRUD) operations, allowing data to be fetched and displayed in the front-end. To create the front-end User interface (UI) mockups were turned into different pages, sharing a uniform design and essential navigation features. Combining the simple and easy-to-use front-end with a responsive back-end the web application fulfills its purpose of supporting study nurses and researchers in managing the research database. Overall, this web application can improve the efficiency of study nurses and can be used to manage data in future studies. While this web application currently only supports one specific period tracker and is tailored to a specific hospital, the framework provided by this project through the use of standards like FHIR and Representational state transfer (REST) is easily expandable in the future. Although the effectiveness of the web application can not yet be quantified, it is safe to assume that it will integrate well into an increasingly digitized workflow.

Contents

List of abbreviations

API application programming interface

CPU central processing unit

CRUD create, read, update and delete

DOM document object model

Express Express.js

FHIR Fast Healthcare Interoperability Resources'

GUI Graphical user interface

HL7 Health Level 7

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

I/O input/output

JS JavaScript

JSON JavaScript Object Notation

Node Node.js

npm Node package manager

OS Operating System

PRO Patient Reported Outcome

RDF Resource Description Framework

REST Representational state transfer

UI User interface

URL Uniform Resource Locator

UKHD Heidelberg University Hospital

Vue Vue.js

XML Extensible Markup Language

List of listings and figures

Listings

List of Figures

List of Tables

1 Introduction

1.1 Subject and motivation

1.1.1 Subject and significance

Sensiplan© is a methodology maintained by the Heidelberg University Hospital (UKHD). Utilizing Patient Reported Outcomes (PROs), fertile days can be accurately determined. This supports users of Sensiplan© in both planned pregnancies as well as natural contraception. The PROs consist of various data points on natural fertility such as basal body temperature, cycle day, cervical mucus, menstruation and intercourse. On account of a new study, a database is to be set up using the FHIR standard. Previously PROs consisted of data exclusively taken from paper forms, whereas PROs now also incorporate data directly sourced from period trackers. The new research database will contain Sensiplan©-compliant PROs from both period trackers and manually collected paper forms. To manage this heterogeneous database, a web application where PROs can be displayed and manually edited is introduced.

1.1.2 Problem

How can a web application be implemented that efficiently supports study nurses and researchers, in managing heterogeneous data from a research back-end, adhering to the following requirements?

- Creating, editing and deleting a participant
- Creating a cycle of Sensiplan© data, editing and deleting the cycle
- Creating a cycle of the period tracker data, editing and deleting the cycle
- Creating a cycle of Sensiplan© and period tracker data, editing and deleting the cycle
- Validation of inputted data

1.1.3 Motivation

Gathering, maintaining and preparing data is an important part of any study. Creating a tool that simplifies data entry and management will streamline data migration from the already existing database to the new research database. This supports study nurses in their administrative responsibilities, improves their efficiency and accuracy, allowing them to spend more time focusing on other duties.

1.2 Challenges

1. As it is not possible to automatically merge the data from the old database with the new one, a tool must be given, so the study nurses are able to manually collect the data for the study in the new database.
2. Moreover, the tool is needed to generally maintain the participant data and delete all records if required.
3. Data manipulation has to be possible, while staying consistent between users.
4. Testing data entry and manipulation.
5. Interoperable solution with FHIR.

1.3 Objectives

1. Development of a web-application to collect cycle data.
2. Mapping of cycle data on Health Level 7 (HL7) FHIR.
3. Implementation of a REST application programming interface (API) to add CRUD-functionalities.
4. Testing and evaluation of the CRUD-functionalities with a FHIR test server.
5. General objectives:
   (a) Usability
   (b) Consistency
   (c) Interoperability
   (d) Maintainability

1.4 Approach

In order to achieve the vision of a consistent and usable web application, a front-end and a back-end are implemented. While both implementations use different technologies, the front-end uses Webstorm as its preferred development environment and the back-end uses VisualStudio code. GitHub is in use as a version control system to ensure persistence of the code[1].

2 Foundation

2.1 Standards

2.1.1 FHIR

FHIR is a free and open source software standard offering high flexibility and scalability published by HL7. FHIRs API is simple and enables interoperability between servers without further configuration and facilitates the connection of multiple servers across different networks. FHIR uses standardised resources to store data in the form of Extensible Markup Language (XML), JavaScript Object Notation (JSON) or Terse Resource Description Framework (RDF) Triple Language. Those resources have a Uniform Resource Locator (URL) to the address, a type, an identified version and contain a set of structured data elements. This set is defined by the resource type. Optional elements and properties have been defined for all resources: an ID, metadata, a base language and a reference to "Implicit Rules". It is possible to link different FHIR resources by their automatically generated ID. FHIR resources are syntactically but not semantically interoperable. To ensure semantic interoperability, it is necessary to describe free text fields by terminologies. To develop the FHIR standard, a RESTful API and several frameworks for messaging, document services and persistent storage and a database need to be implemented.[2]

2.1.2 Sensiplan©

Sensiplan© is a method for visualizing and interpreting the individual's menstrual cycle through fertility observation and intensive body perception. Used correctly it can be as effective as the pill after three cycles, making it a safe non-hormonal and less expensive option to avoid or even achieve conception[3]. Everything the participants need to use Sensiplan© is a cycle form and a thermometer. Some of Sensiplan's© benefits include the ability to be started directly after hormonal contraception as well as increasing the awareness of partnership responsibility in family planning. Some downsides compared to other methods of family planning are the time needed to track data points daily and the impact an unhealthy lifestyle can have on the overall effectiveness of Sensiplan©.[4]

2.2 Technologies

2.2.1 Axios

Axios is a promise-based HTTP client for Node.js (Node) and web browsers. It is isomorphic, running in both browser and Node from the same codebase. On the server side it uses acpNode native HTTP module, using XML HTTP Requests on the client/browser side. Axios intercepts request and response

date, transforms them and aborts requests. The transformations are performed directly as JSON data. It can also be used to protect the client side from Cross-Site-Request-Forgery. Axios also supports the JavaScript (JS) promise API.[5]

2.2.2 Docker

Docker is a tool designed to solve the problem of 'dependency hell' caused by various runtime environments on different Operating Systems (OSs), by creating containers that run all application dependencies on the Docker engine. This forms the basis for many cloud-native applications, helps to develop ideas quickly and securely, supports all types of applications and allows them to run on different operating systems. Docker is supported by an integrated BuildKit, which provides architectural enhancements to create faster, more accurate and portable applications. DockerCLI simplifies container management through a set of commands.[6]

2.2.3 Express.js

Express.js (Express) Express is a minimal and flexible web application framework that enhances Node. It extends Nodes core functionalities by providing a robust set of web application functions. Additionally, Express facilitates developing fast and simple APIs that support multiple HTTP service methods and middleware.[7]

2.2.4 Node.js

Node is an open source development platform built to run server-side JS code. In most applications Node runs on a dedicated Hypertext Transfer Protocol Secure (HTTPS) server. Node executes JS requests asynchronously on a single thread, meaning that all operations are bundled into a single stack that processes events successively. To make that single stack as efficient as possible, Node implements callbacks. Callbacks are functions that are called at completion in contrast to synchronous functions that are called and executed sequentially. This enables Node to run other code in the meantime, allowing as many input/output (I/O) operations as the OS can handle. Since JS itself is single-threaded Node uses the system kernel to execute processes whenever possible. With virtually all OSs being multi-threaded, they can handle multiple operations simultaneously. Once an operation is completed, the assigned callback is returned to Node where it is placed on top of the stack to eventually be executed. Node avoids applications slowing down or even crashing due to long central processing unit (CPU) cycles by increasing the number of processes in favor of more complex ones. The asynchronous structure essentially prevents blocking the I/O queue, making Node great at executing JS at any scale.[8]

2.2.5 Vue.js

Vue.js (Vue) Vue is an open source library for creating interactive web interfaces. It can be effortlessly integrated into preexisting projects, by only providing the user interface/view layer. Vue is versatility makes it a useful tool for Single-page applications, Full-stack and Server-side rendering, static site generation, desktop, mobile, and even terminal clients. Vue uses reactive data-binding to achieve a data-driven view in the document object model (DOM). Put simply, instead of updating the DOM directly data is bound to HyperText Markup Language (HTML) templates via Vues custom syntax. When data is updated changes are initially made to a virtual DOM. The virtual DOM is then compared against the real DOM, rendering the differences. This allows for more updates at faster rates compared to updating the entire DOM. Vues component system is another core feature, adding another layer of abstraction.[9]

2.2.6 Vuetify

Vuetify is a Material Design component framework for Vue[10]. It provides a variety of UI components and features like a grid system, typography, and pre-designed UI elements such as buttons, navigation, and form controls. One of the key benefits of using Vuetify is the consistency it provides across different devices through the use of responsive design principles. Vuetify ensures that the layout of the UI automatically adjusts to fit the size of the screen it is being displayed on.[11]

2.2.7 Vue CLI

Vue CLI is a baseline tool for Vue. It tracks different build tools present in the Vue projects, streamlining their configurations and permitting changes without having to eject the web application. It is highly customizable with plugin support for Node package manager (npm) packages like Babel/TypeScript transpilation, ESLint integration, unit testing, and end-to-end testing.[12]

2.2.8 REST

REST is an architectural design, based on CRUD Methods, to build web services which allow creation of distributed systems. It is based on the idea of separating the client and server into distinct components, with the client responsible for making requests to the server and the server responsible for processing those requests and returning a response. RESTful web services follow a set of guidelines, known as the REST architectural style,which defines how the client and server should interact. These guidelines include the use of HTTP verbs (such as GET, POST, PUT, and DELETE) to represent different actions, the use of URLs to identify resources, and the use of HTTP

status codes to indicate the success or failure of a request. RESTful web services also use a standard data format, such as JSON or XML, to exchange information between the client and server. This allows for interoperability between different systems and allows developers to easily create and consume web services. Overall, REST architectural design and RESTful web services provide a simple and scalable way to build distributed systems and enable the creation of powerful APIs for a wide range of applications.[13]

3 Materials and methods

After analysing Sensiplan© and period tracker form as well as the questionnaire, the project was split into front-end and back-end. The FHIR standard was chosen, due to its interoperability and the projects medical context. The communication between front-end, back-end and a FHIR server is realised by HTTP methods.

3.1 Back-end

The back-end is implemented to communicate with both the front-end and a FHIR server using the following technologies:

- Axios version: 1.2.1
- Express version: 4.18.2
- Node version: 14.17.1
- Docker version: 20.10.21
- Visual Studio Code version: 1.74.2
- FHIR server version: 6.3.2-SNAPSHOT/312128754b/2022-11-30

Approach

For the realisation of the back-end, a FHIR test server was specified that has at least version 6.3. Based on the questionnaire criteria, *Patient* and *QuestionnaireResponse* were selected from the list of FHIR resources. A mapping process was then carried out to connect the Sensiplan© and Period Tracker forms to the selected FHIR resources. As a consequence, the respective JSON schema was evaluated in order to then fit the given questionnaire into the FHIR resource JSON schema. After setting a foundation, the implementation of the REST interface was started, in the following order:

1. HTTP-POST: ability to add a new *Patient* and *QuestionnaireResponse*
2. HTTP-GET: ability to retreive already added ressources.
3. HTTP-DELETE: ability to delete earlier added ressources.
4. HTTP-PUT: ability to alter added ressources.

Testing of the REST interface was performed to ensure that all criteria were met. A Docker container is used to host the back-end, which is executed via Docker.

FHIR test-server

A public FHIR test server is used in this prototype [14]. The FHIR server offers various resources, like *Questionnaire*, *Patient*, *Practitioner*, *QuestionnaireResponse*, etc. for storing (medical) data. However, only *Patient* and *QuestionnaireResponse* are used in this web application.

Patient resource

The FHIR resource *Patient* contains all relevant data for a given participant. Relevant data such as administrative, demographic and identifying data is stored in a *Patient* object.[15]

```
{
  "resourceType" : "Patient",
  "identifier" : [{ Identifier }],
  "active" : <boolean>,
  "name" : [{ HumanName }],
  "telecom" : [{ ContactPoint }],
  "gender" : "<code>",
  "birthDate" : "<date>",
  "deceasedBoolean" : <boolean>,
  "deceasedDateTime" : "<dateTime>",
  "address" : [{ Address }],
  "maritalStatus" : { CodeableConcept }
  "multipleBirthBoolean" : <boolean>,
  "multipleBirthInteger" : <integer>,
  "photo" : [{ Attachment }],
  "contact" : [{
    "relationship" : [{ CodeableConcept }],
    "name" : { HumanName },
    "telecom" : [{ ContactPoint }],
    "address" : { Address },
    "gender" : "<code>",
    "organization" : { Reference(Organization) },
    "period" : { Period }
  }],
  "communication" : [{
    "language" : { CodeableConcept },
    "preferred" : <boolean>
  }],
  "generalPractitioner" : [{
        Reference(Organization|Practitioner|
    PractitionerRole) }],
  "managingOrganization" : {
        Reference(Organization) },
  "link" : [{
    "other" : { Reference(Patient|RelatedPerson) },
    "type" : "<code>" }]
}
```

Listing 1: Patient.json

QuestionnaireResponse resource

The FHIR resource *QuestionnaireResponse* contains individual answers of participants to a questionnaire. This resource is split in two parts: Metadata like author or status and participant data in one or several item arrays.[16]

```
{
  "resourceType" : "QuestionnaireResponse",
  "identifier" : [{ Identifier }],
  "basedOn" : [{
        Reference(CarePlan|ServiceRequest) }],
  "partOf" : [{ Reference(Observation|Procedure) }],
  "questionnaire" : "<canonical(Questionnaire)>",
  "status" : "<code>",
  "subject" : { Reference(Any) },
  "encounter" : { Reference(Encounter) },
  "authored" : "<dateTime>",
  "author" : { Reference(Device|Organization
        |Patient|Practitioner|
    PractitionerRole|RelatedPerson) },
  "source" : { Reference(Device|Organization
        |Patient|Practitioner|
    PractitionerRole|RelatedPerson) },
  "item" : [{ "linkId" : "<string>",
    "definition" : "<uri>", "text" : "<string>",
    "answer" : [{ "valueBoolean" : <boolean>,
      "valueDecimal" : <decimal>,
      "valueInteger" : <integer>,
      "valueDate" : "<date>",
      "valueDateTime" : "<dateTime>",
      "valueTime" : "<time>",
      "valueString" : "<string>",
      "valueUri" : "<uri>",
      "valueAttachment" : { Attachment },
      "valueCoding" : { Coding },
      "valueQuantity" : {
        Quantity(SimpleQuantity) },
      "valueReference" : { Reference(Any) },
  "item" : [{
    Content as for QuestionnaireResponse.item }]}],
  "item" : [{
    Content as for QuestionnaireResponse.item }]}]
}
```

Listing 2: QuestionnaireResponse.json

3.2 Front-end

The front-end was implemented using a combination of frameworks. The following components and its versions were configured to meet the requirements of this particular project:

- Vue version: 2.6.14
- Vuetify version: 2.6.0
- Vue-axios version: 3.5.2
- Vue-router version: 3.5.1
- Axios version: 1.1.3
- Core-js version: 3.8.3
- Vue-cli-plugin-vuetify version: 2.5.8
- Node version: 19.1.0
- Docker version: 20.10.21
- Including babel of versions 7.12.16 and the corresponding Vue compatibility of versions 5.0.0
- Webstorm version: 2022.2.3

After the analysis of the given data mockups were created to evaluate and design a Graphical user interface (GUI) for data collection and management. Firstly, a basic design was needed, therefore, the following aspects were determined:

1. Set an appropriate colour scheme
2. Set a global design for every page, this includes "Header" / "Navigationbar", "Return button", "Font", "Font size" and "Input screen"
3. What pages are needed to fullfill the requirements
4. How pages connect / interact with each other
5. How to display error messages

Colour scheme

The evaluation of the colour scheme included different period trackers [17], the colour scheme of the corresponding clinic and overall well designed web applications [18].

Global design

The decision was made to include a global header / navigation-bar for every single page. Additionally functions like returning to last page, display of multiple objects (Patients, Periods) and design of the input screen for the questionnaires need to be harmonized, in order to simplify the web application.

Pages

In order to provide the needed functionalities the following pages are needed:

1. Homepage
2. Single patient overview using CRUD
3. Single questionnaire overview using CRUD

Connection between pages

After determining the needed pages, a startpoint for the application was defined to be the homepage. From there one should be able to access the patient and questionnaire overview. Furthermore, due to logical restrictions it should not be possible to access the questionnaire overview from the patient overview.

Display errors

It is crucial to display occurring errors to the user. This should be done in a easy to understand and harmonized way. Therefore, a generic error message should be created.

To fulfill the criteria mockups were created to evaluate the prototype GUI. Following the mockups as the fundamental structure the front-end was implemented and tested. After implementing both back-end and front-end the connection between these two was achieved via HTTP methods and tested.

Figure 1: Mockup homepage

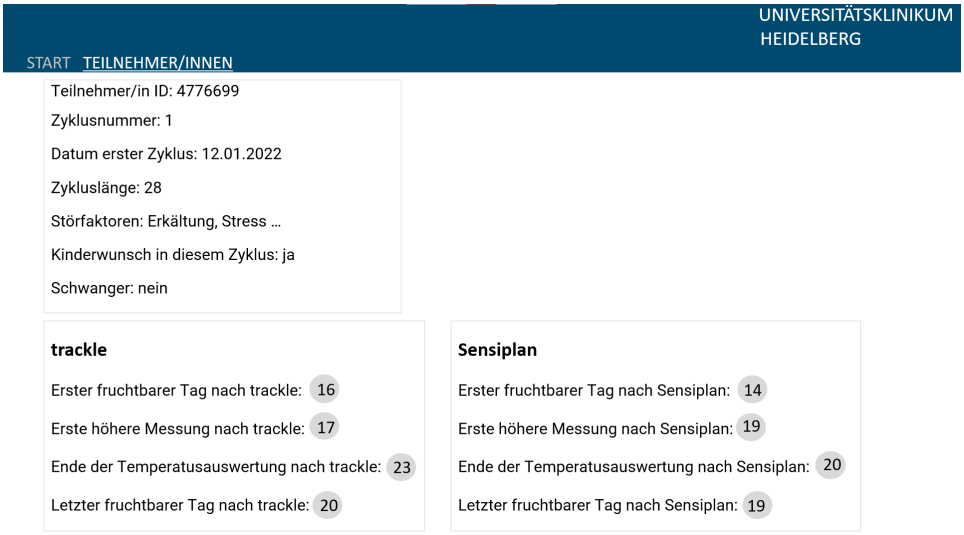

Figure 2: Mockup single cycle page

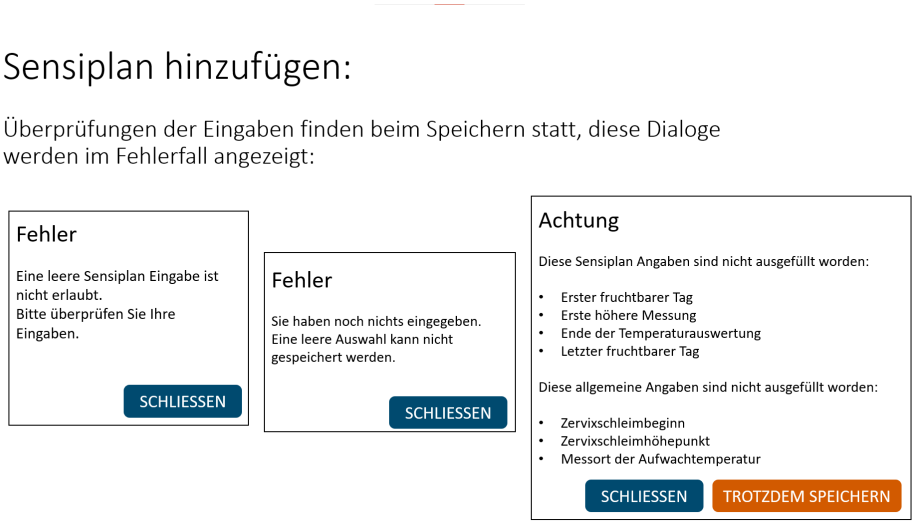

Figure 3: Mockup Sensiplan© error dialogues

4 Results

4.1 Back-end

The back-end functions as middle-ware between the front-end and the FHIR test server. It provides a REST interface for the front-end. After processing an incoming request, a new request for the FHIR test server's REST interface is created in the back-end.

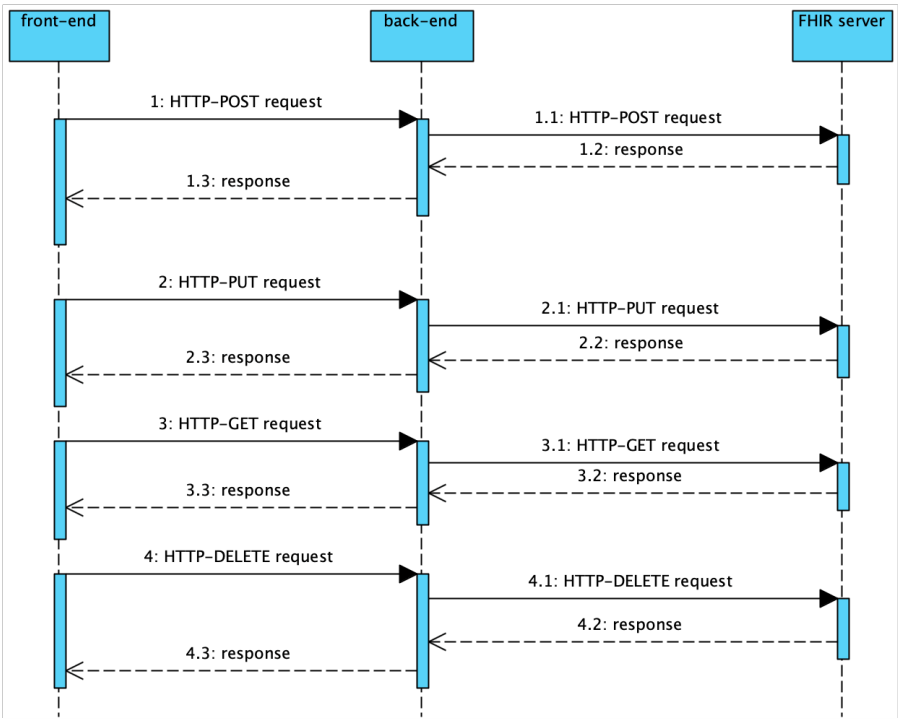

Figure 4: REST Structure

REST Interface

The REST interface supports the FHIR resources *Patient* and *QuestionnaireResponse*. Both FHIR resource types, have a dedicated handler that uses the HTTP protocol to support full CRUD functionality. Each CRUD method has the same structure: address, header, body and return along with a status code of 200, 201 or 400. The incoming data from the front-end is processed according to the respective FHIR resource schema and then sent to the FHIR test server without further validation or processing. The FHIR test server, on the other hand, validates all incoming resources referring to the given FHIR resource schema and returns an error to the back-end if the request is not schema-compliant. Two types of error can occur in these processes:

1. The request from the front-end to the back-end is not valid, e.g. missing variables.
2. The FHIR test-server validates the incoming *Patient*/*QuestionnaireResponse* resource as a variable type, not being schema conform.

In the event of any error, the status code 400 is returned to the front-end as response to the request. If there is no data for a mandatory variable, they must be set to -1 or "-1". The HTTP-PUT method overwrites the existing data within a resource if it already exists. If no resource exists for the passed ID, a new one is created instead.

Patient **handler**

- Create
  - Create a *Patient* resource with the HTTP-POST method
    - Address: localhost:3000/patient
    - Header: empty
    - Body: "idTrackle" (String), *idSensiplan* (String), "idOldDB" (String)
    - Return: 201 and created *Patient* resource if successful, 400 if IDs are missing
- Read
  - Get a *Patient* resource with the HTTP-GET method
    - Address: localhost:3000/patient/:id
    - Header: *idPatient* (String)
    - Body: empty
    - Return: 200 and *Patient* resource if successful, 400 if *Patient* does not exist

  - Get all linked *QuestionnaireResponse* IDs for a given *Patient* resource
    * Address: localhost:3000/patient/:id/everything
    * Header: *idPatient* (String)
    * Body: empty
    * Return: 200 and JSON object with all corresponding *QuestionnaireResponse* IDs if successful

- Delete
  - Delete a *Patient* resource with the HTTP-DELETE method
    * Address: localhost:3000/patient/:id
    * Header: *idPatient* (String)
    * Body: empty
    * Return: 200 if successful, 400 if *Patient* does not exist and 409 if *Patient* has *QuestionnaireResponse* resources linked

- Update
  - Update a *Patient* resource with the HTTP-PUT method
    * Address: localhost:3000/patient/:id
    * Header: *idPatient* (String)
    * Body: *idTrackle* (String), *idSensiplan* (String), *idOldDB* (String)
    * Return: 201 and modified *Patient* if successful, 400 if IDs are missing

***QuestionnaireResponse* handler**

- Create:
  - Create a *QuestionnaireResponse* resource for Sensiplan© with a HTTP-POST method
    * Address: localhost:3000/QuestionnaireResponse/Sensiplan
    * Header: empty
    * Body: *idPatient* (String) , questions 2 – 7_3, 9,11,13,15 and 16 – 19
    * Return: 201 and created *QuestionnaireResponse* resource if successful, 400 if missing fields

– Create a *QuestionnaireResponse* resource for a period tracker with a HTTP-POST method
  * Address: localhost:3000/QuestionnaireResponse/trackle
  * Header: empty
  * Body: *idPatient* (String) , questions 2 – 7_3, 8,10,12,14 and 16 – 19
  * Return: 201 and created *QuestionnaireResponse* resource if successful, 400 if missing fields

Question number	**Type**	**Question**
idPatient	Int	-
2	Int	Cycle number
3	Date	First cycle day
4	Int	Cycle length
5	String	Disturbing factor
6	String	Childbearing preferences
7	String	Pregnant
Only 7 when yes		
7_1	Boolean	Test - positive?
7_2	Boolean	Prolonged high temperature
7_3	String	Ausscheider?
8	Int	First fertile day (Trackle)
9	Int	First fertile day (Sensiplan©)
10	Int	First high temperature (Trackle)
11	Int	First high temperature (Sensiplan©)
12	Int	End of temperature evaluation (Trackle)
13	Int	End of temperature evaluation (Sensiplan©)
14	Int	Last fertile day (Trackle)
15	Int	Last fertile day (Sensiplan©)
Intercourse per cycle day		
16_1	String	No intercourse
16_2	String	Unprotected intercourse
16_3	String	Intercourse with condom
16_4	String	Intercourse with another contraceptive
16_5	String	No data
17	String	Measuring location(wake up temperature)
18	Int	begin cervical mucus
19	String	Cervical mucus peak

Table 1: Patient questions

- Read
  - Get a *QuestionnaireResponse* resource with a HTTP-GET method
    - Address: localhost:3000/QuestionnaireResponse/:id
    - Header: *idQuestionnaireResponse* (String)
    - Body: empty
    - Return: 200 and *QuestionnaireResponse* resource if successful, 400 if *QuestionnaireResponse* does not exist
- Delete
  - Delete a *QuestionnaireResponse* resource with a HTTP-DELETE method
    - Address: localhost:3000/QuestionnaireResponse/:id
    - Header: *idQuestionnaireResponse* (String)
    - Body: empty
    - Return: 200 if successful, 400 if *QuestionnaireResponse* does not exist
- Update
  - Update a *QuestionnaireResponse* resource for Sensiplan© with a HTTP-PUT method
    - Address: localhost:3000/QuestionnaireResponse/Sensiplan/:id
    - Header: *idQuestionnaireResponse* (String)
    - Body: *idQuestionnaireResponse* (String), *idPatient* (String) , questions 2 – 7_3, 9,11,13,15 and 16 – 19
    - Return: 201 and create a *QuestionnaireResponse* resource if successful, 400 if fields are missing
  - Update a *QuestionnaireResponse* resource for a period tracker with a HTTP-PUT method
    - Address: localhost:3000/QuestionnaireResponse/trackle/:id
    - Header: *idQuestionnaireResponse* (String)
    - Body: *idQuestionnaireResponse* (String), *idPatient* (String), questions 2 – 7_3, 8,10,12,14 and 16 – 19
    - Return: 201 and create a *QuestionnaireResponse* resource if successful, 400 if fields are missing

4.2 Front-end

The UKHD colour scheme (Hexadecimal: 004a6f) was chosen for the front-end, after various colour and component combinations have been tested and compared. The font: "Material Design spec Roboto Font" with size 12 was primarily used. The basic structure of the application provides a minimal design with a navbar leading to pages related to the participants as well as a start page with a logo. To improve user workflow, routing and inconsistencies during returning to previous pages, cancel and back buttons were added. The created pages are as following:

- a homepage with a logo
- a participants page with the IDs (the current database, Sensiplan©, period tracker and of the old database)
- a participant page with all IDs
- a create participant page
- an edit participant page
- a cycles page with all cycles of the selected participant (with the cycle number, Sensiplan© and/or period tracker IDs)
- an add cycle page with basic information to the cycle
- some following pages to the add cycle page for adding a period tracker and/or Sensiplan© cycle
- a single cycle page with both Sensiplan© and period tracker data
- an editing page of cycle data with both Sensiplan© and period tracker data

The following components have been selected from the Vuetify range: v-cards, number textfields, a date picker component, a navigation bar, v-data-table(s), v-select components and v-chip components. The start page can be accessed via the navbar. It consists of a welcome message for the users and a generated logo of the project.

Figure 5: Homepage

The participants page consists of a table presenting data such as *idSensiplan*, *idTrackle*, *idOldDB* and *idPatient*. In order to avoid overloading the front-end, with FHIR created participants, only three participants were hardcoded into the server request. The site has several functions including the ability to search for IDs within the table and sort columns, a button to access a page for adding new participants, and the ability to click on a row to open a participant view with a router link.

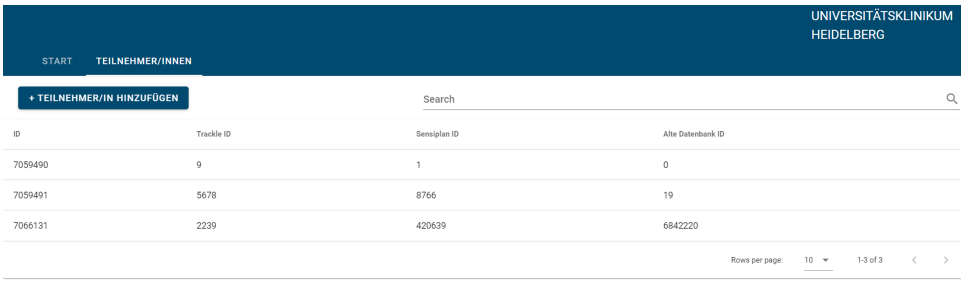

Figure 6: Participants page

The participant view, which is accessed through the router, shows the four IDs and includes four buttons: showing the cycle site for the participant through a router link, going back to the participants table, editing the participant, and deleting the participant.

Figure 7: Participant page

A participant can be added by adding the *idtrackle*, *idSensiplan* and/or *idOldDB*.

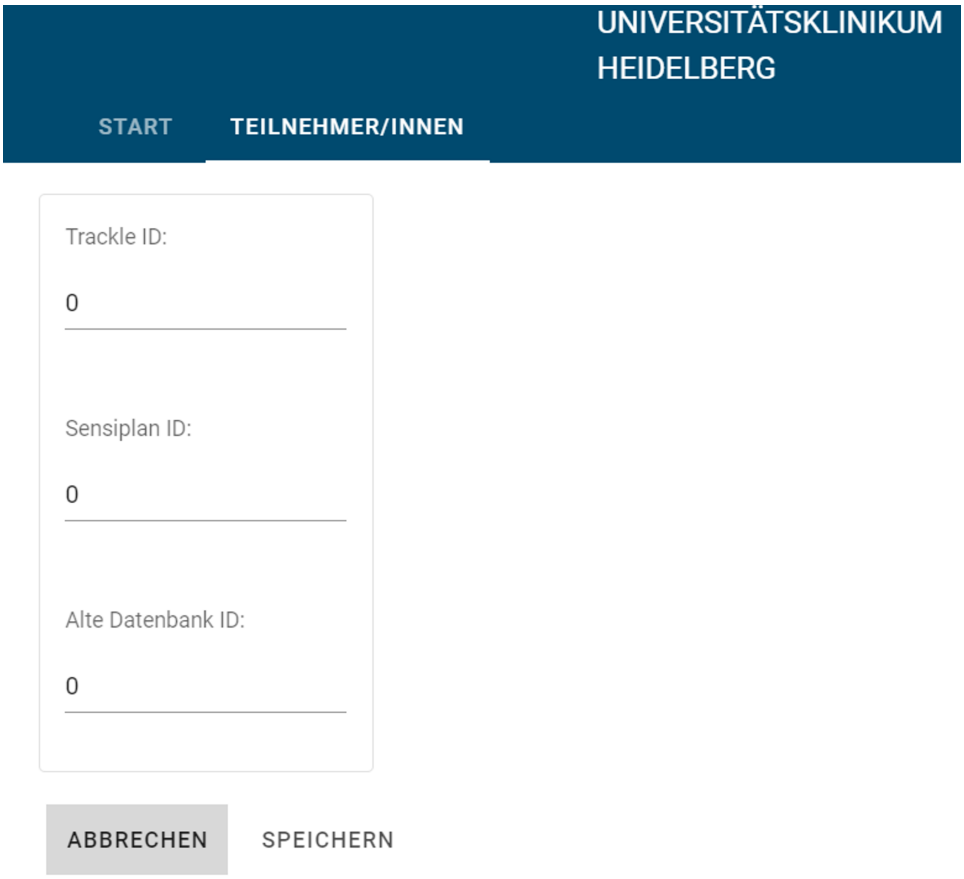

Figure 8: Create participant

Validation of input data includes checking negative ID values, blank entries as well as permissible ID combinations, such as *idOldDB* must be combined with either *idSensiplan* or *idTrackle*. Dialogues with detailed information about the incorrect inputs are the following:

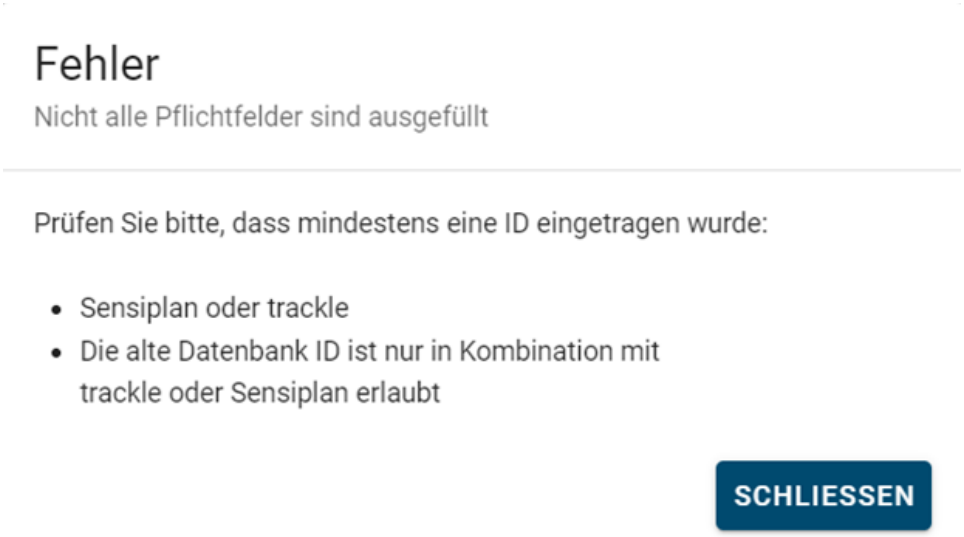

Figure 9: Dialogue nothing filled, *Participant pages*

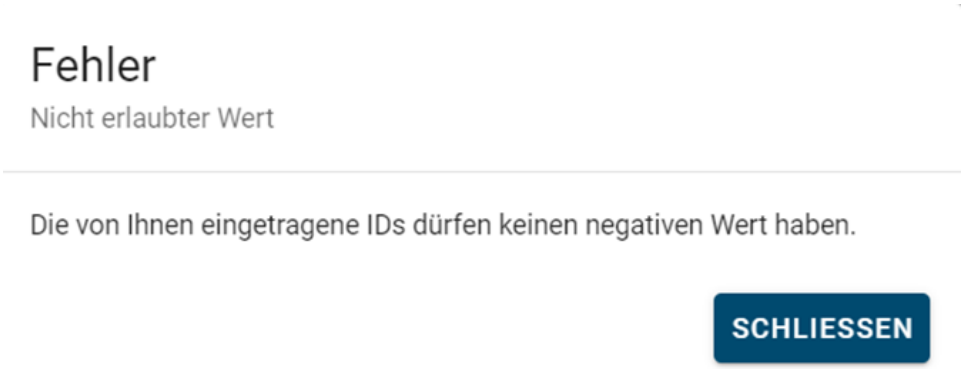

Figure 10: Dialogue negative ID, *Participant pages*

After saving the participant data can be edited. The validation of edited IDs corresponds with the create participant validations and also includes a verification of changes, so that unchanged data is not sent to the back-end.

Figure 11: Edit participant page

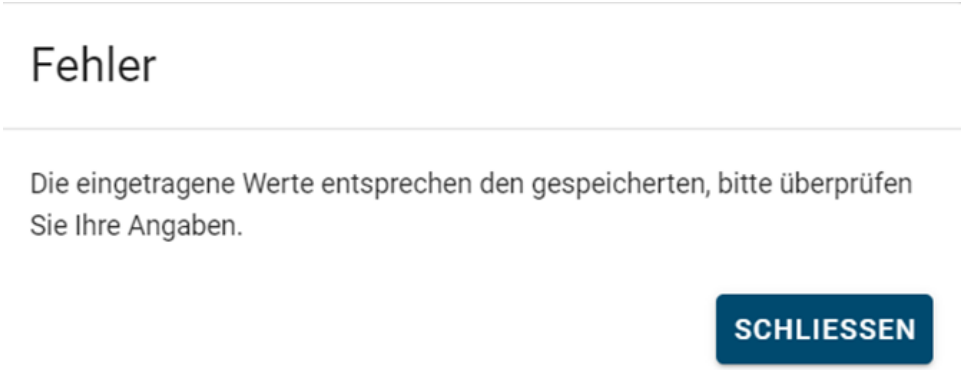

Figure 12: Dialogue no changes, *Edit participant page*

The Cycles page is similar to a Participant page. However, it contains the cycle number and the corresponding period *idTrackle* and *idSensiplan* if this cycle contains both, otherwise a zero is displayed, allowing filtering by Sensiplan© or Trackle.

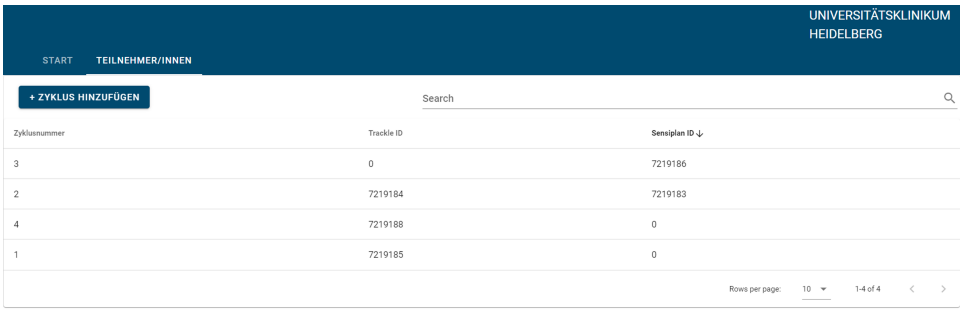

Figure 13: Cycles page

A new cycle can be added with the following information: cycle length, cycle days, intercourse, fertile days, and temperature. The user can then choose the cycle type (period tracker and or Sensiplan©) depending on the information collected from the participants and is transferred to that page.

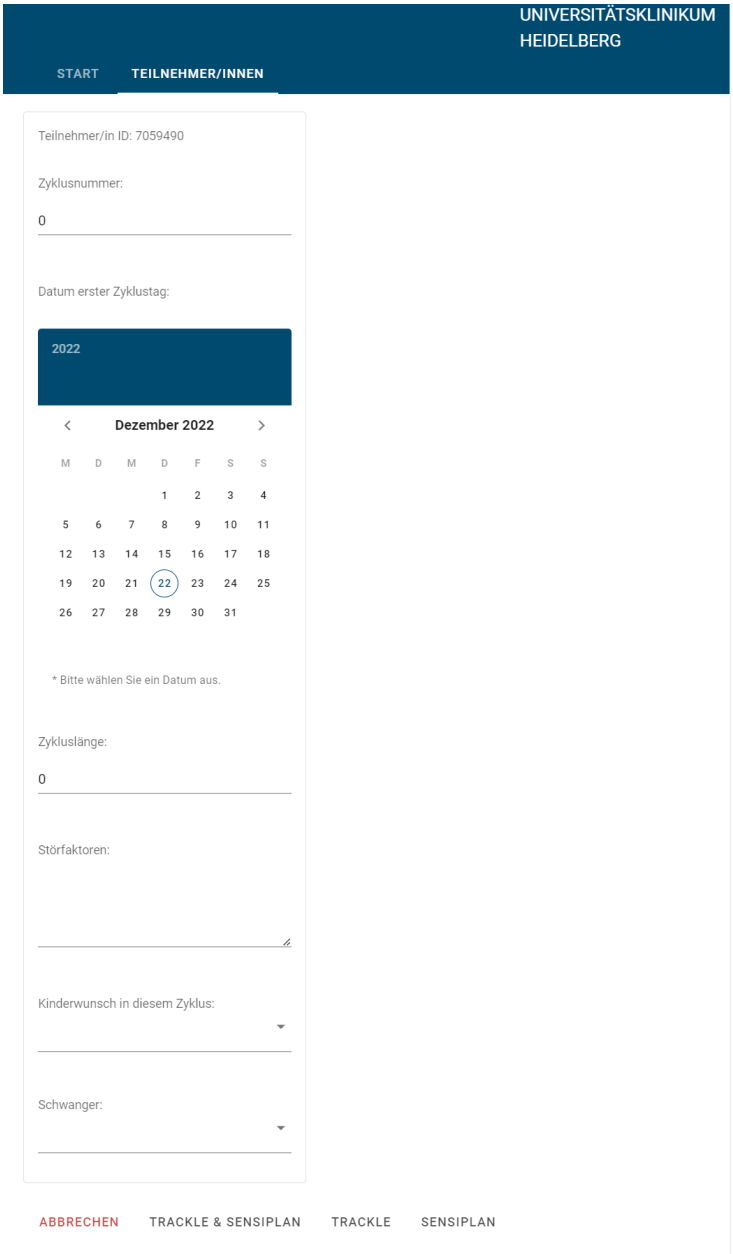

Figure 14: Add cycle

To create a valid cycle on the add cycle page, all entries except for the disturbance factor must be filled in. If the cycle length and number are incorrect, an error dialogue is displayed.

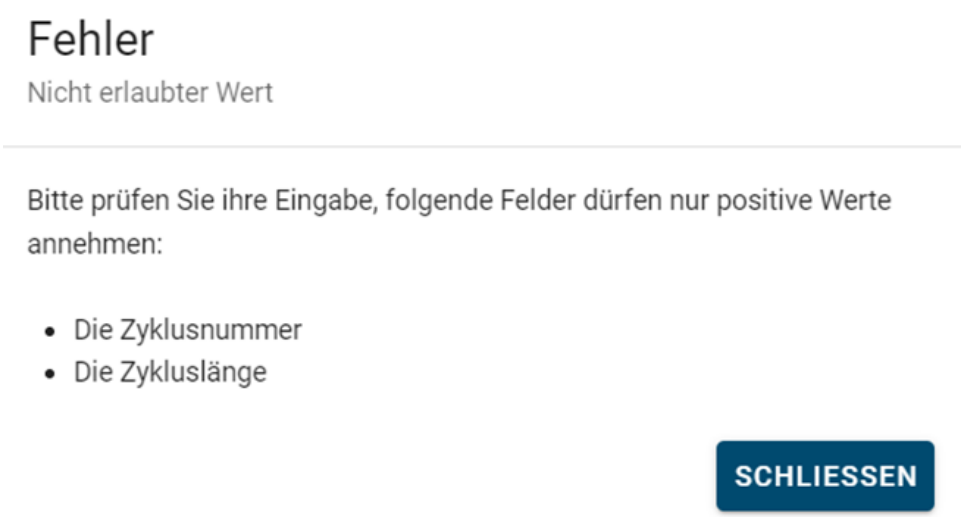

Figure 15: Dialogue numbers negative or not filled, *Add cycle page*

In case the cycle number and length are correct, the following dialogue is displayed:

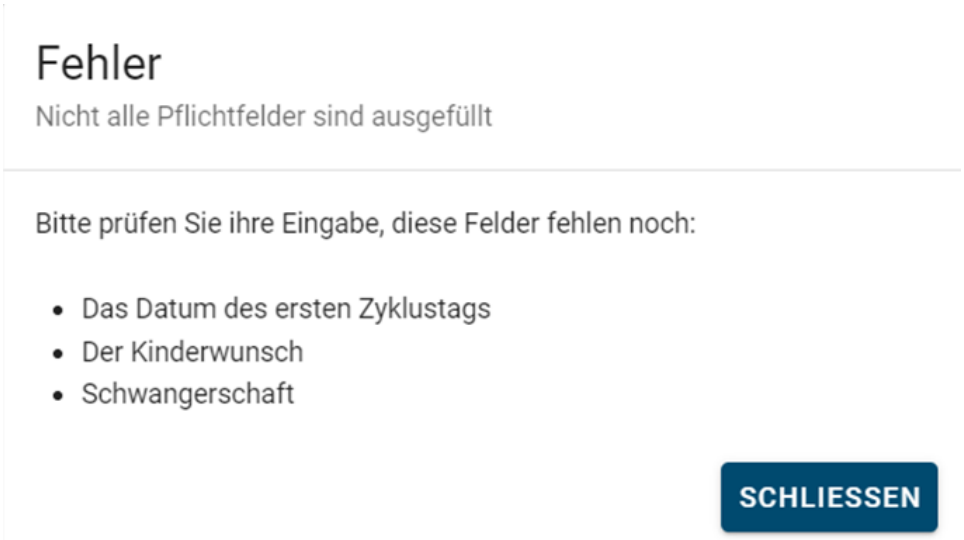

Figure 16: Dialogue not everything is filled, *Add cycle page*

Following the add cycle page a Sensiplan© and/or a period tracking cycle can be created. To select a specific cycle day, v-select is used.

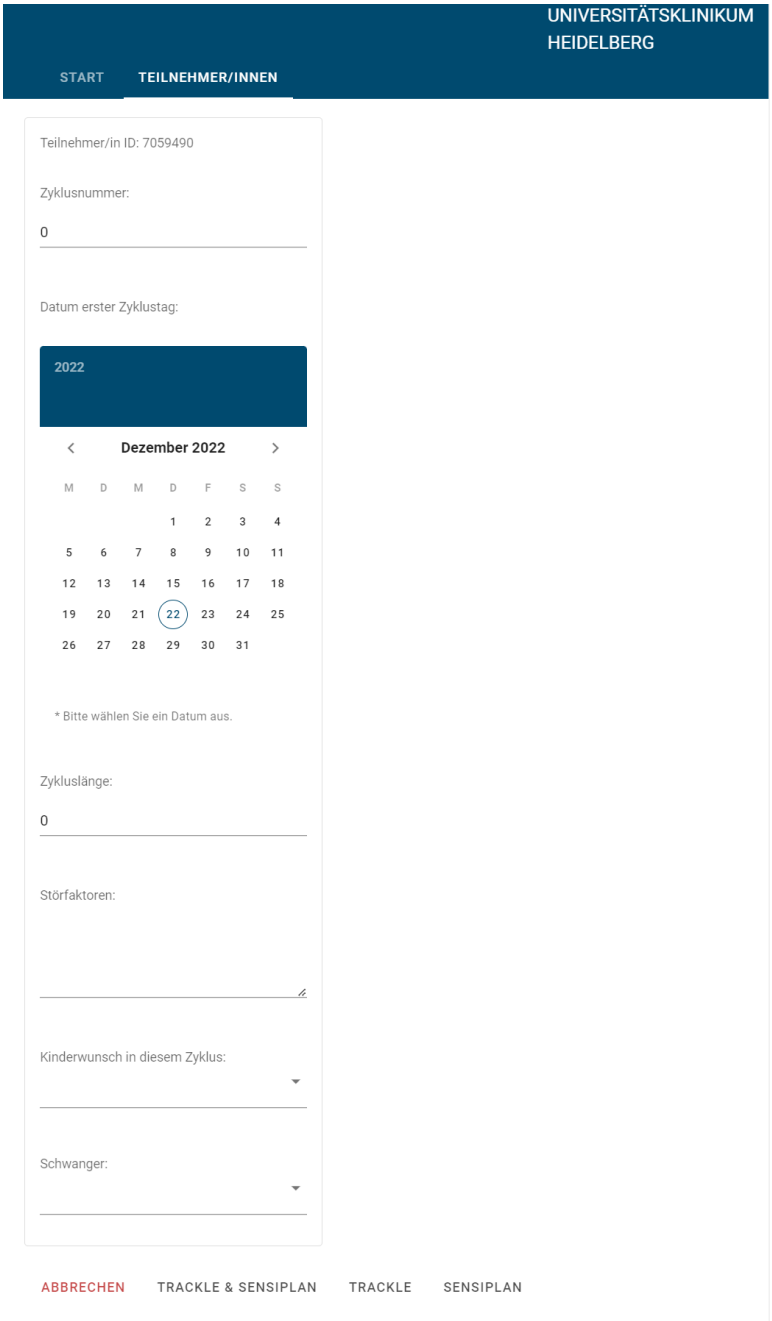

Figure 17: Add cycle, *first page*

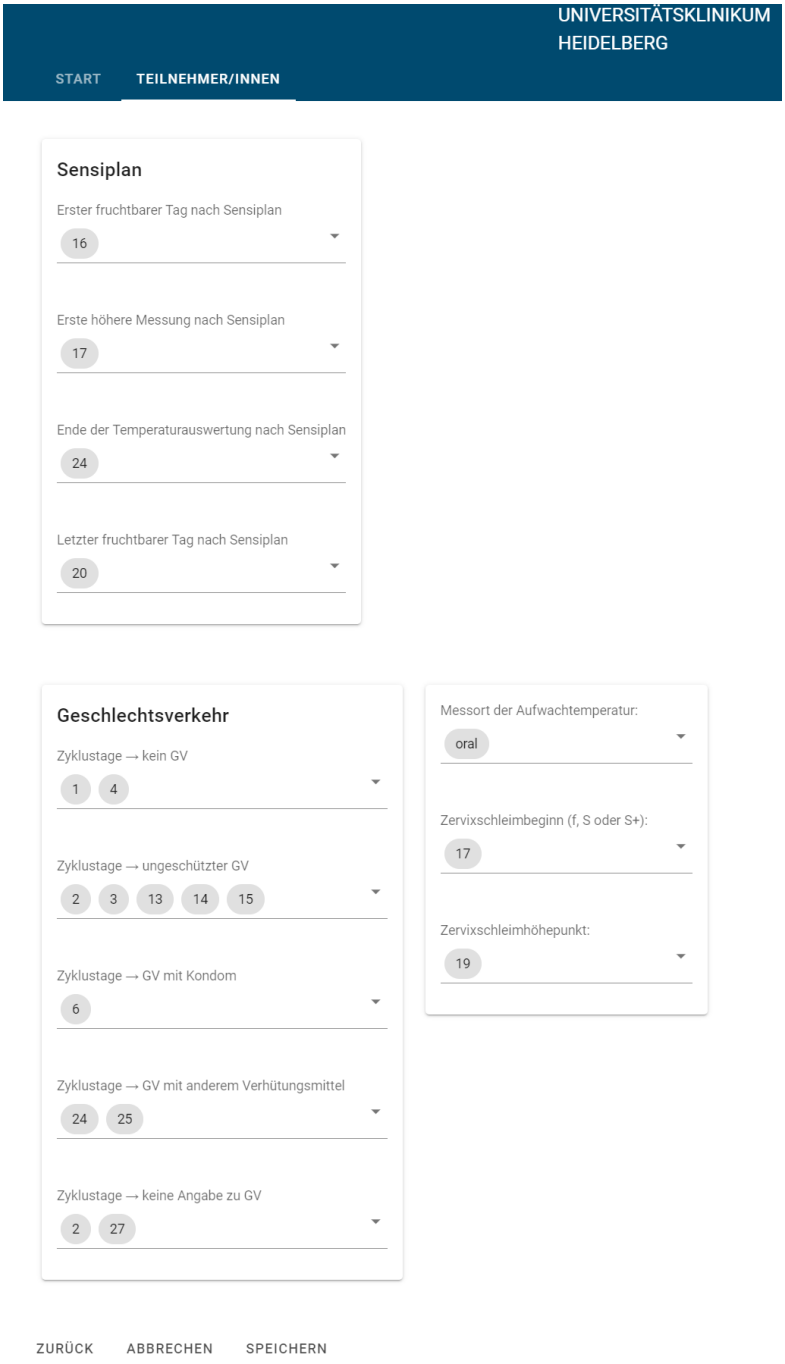

Figure 18: Add cycle Sensiplan©

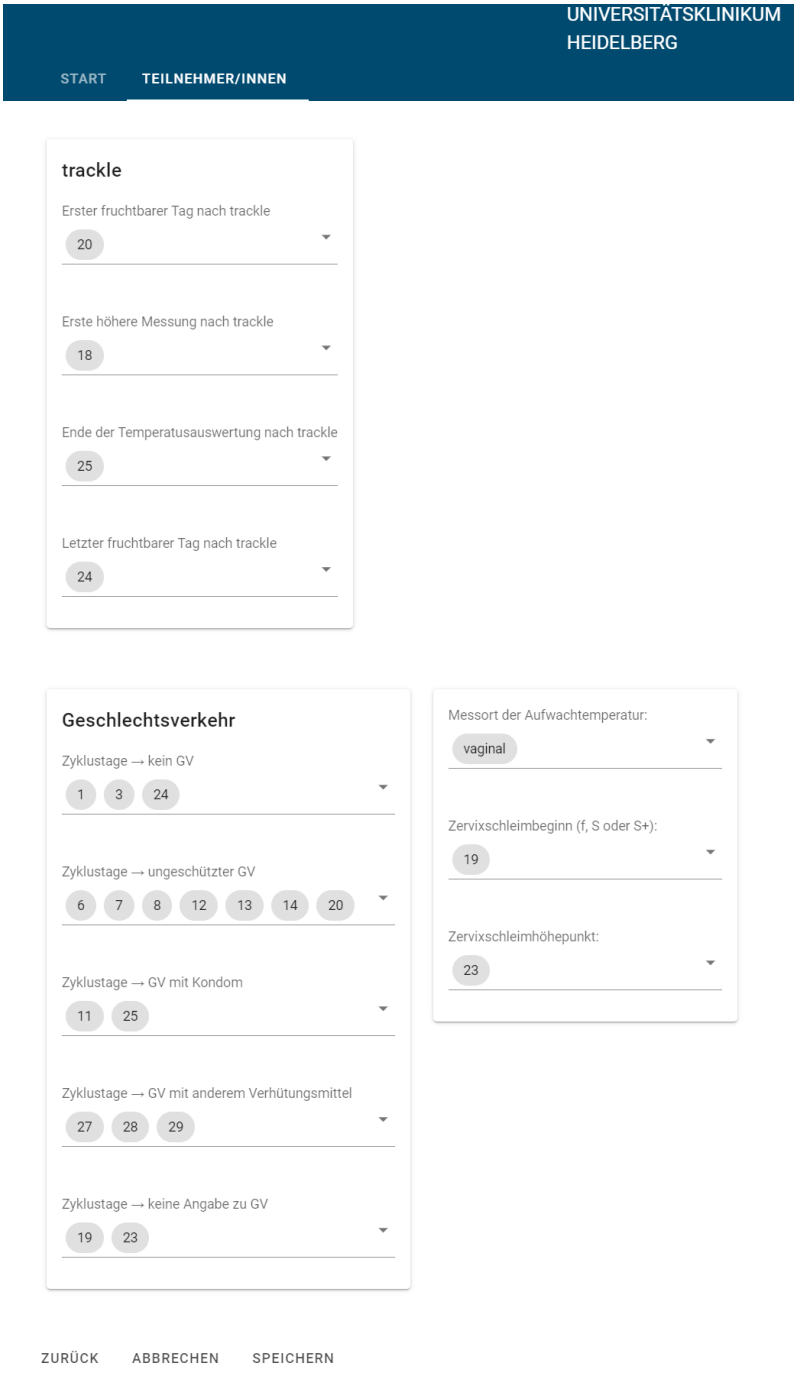

Figure 19: Add cycle period tracker

In order to create a valid Sensiplan© or period tracker cycle, at least one entry must be filled in, in which case a dialogue with all missing entries is displayed and the user can decide whether to save the process or not. If no entry is filled in, a dialogue informs the user that the process will not be saved due to missing entries.

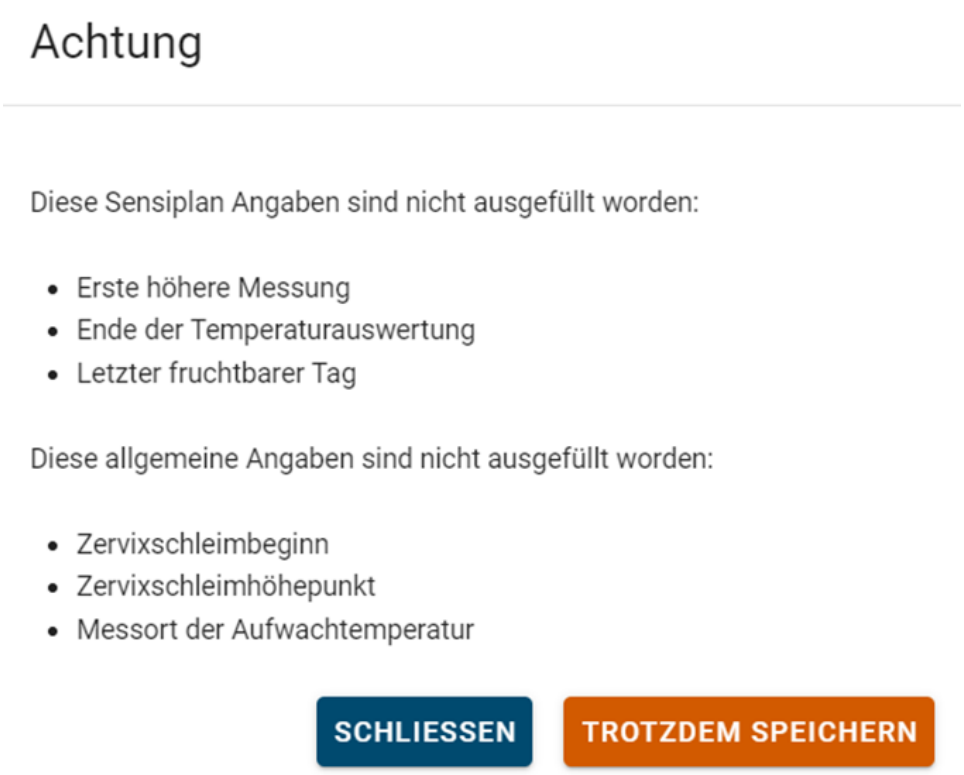

Figure 20: Add cycle Sensiplan©, period tracker or both, *incomplete entries*

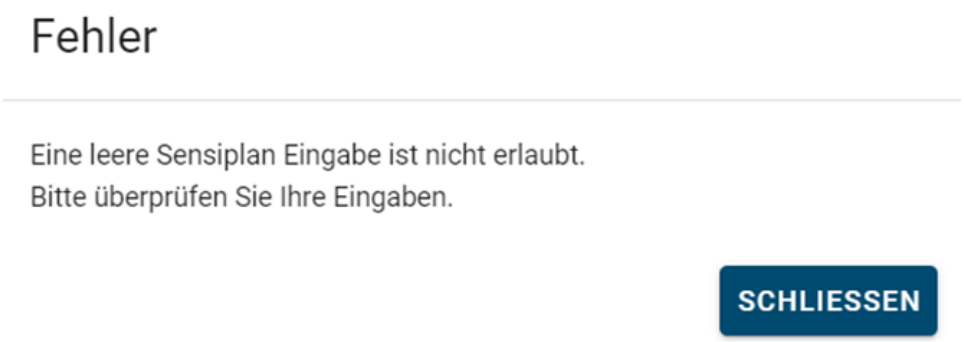

Figure 21: Add cycle Sensiplan©, *no entries*

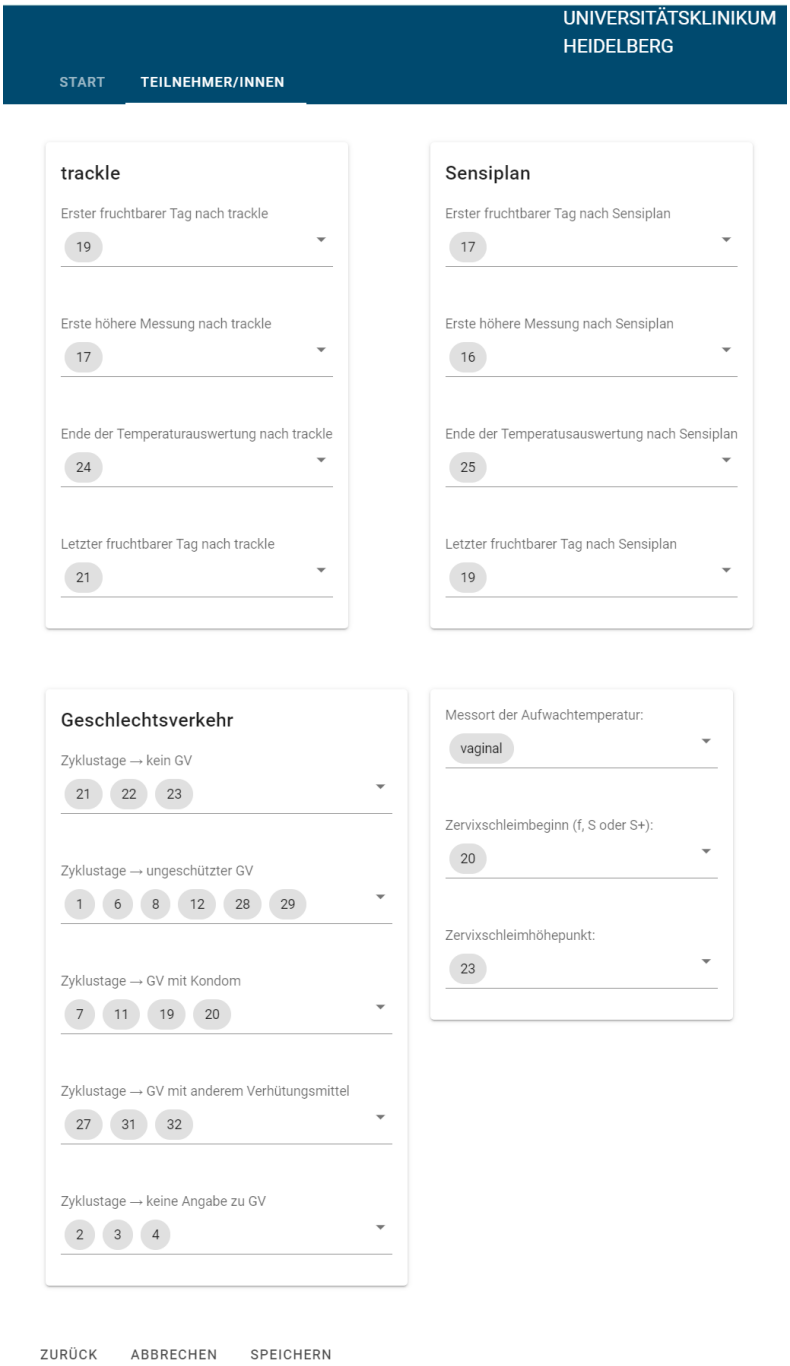

Figure 22: Add cycle Sensiplan© and period tracker

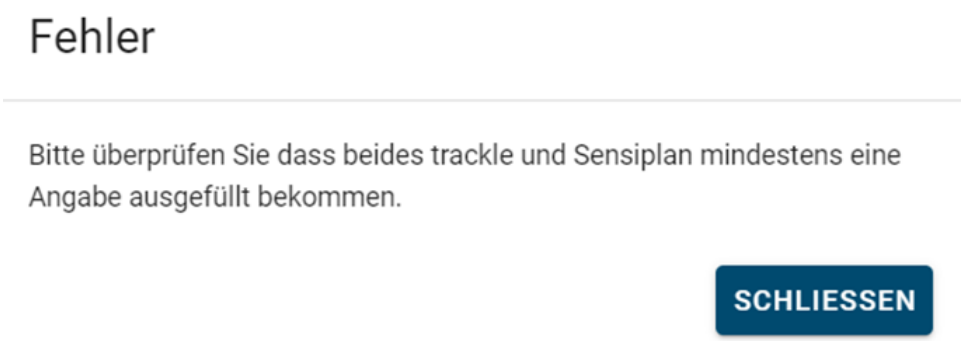

Figure 23: Add cycle Sensiplan© and period tracker, *not filled*

The cycle page includes general cycle information at the top and middle sections consisting of side-by-side views of the period tracker and Sensiplan© data, with zero values if not present. The bottom of the cycle single view contains information about intercourse and general data points. It also includes tree buttons to edit or delete the cycle and also to return to the cycles page. The page is accessed through the router.

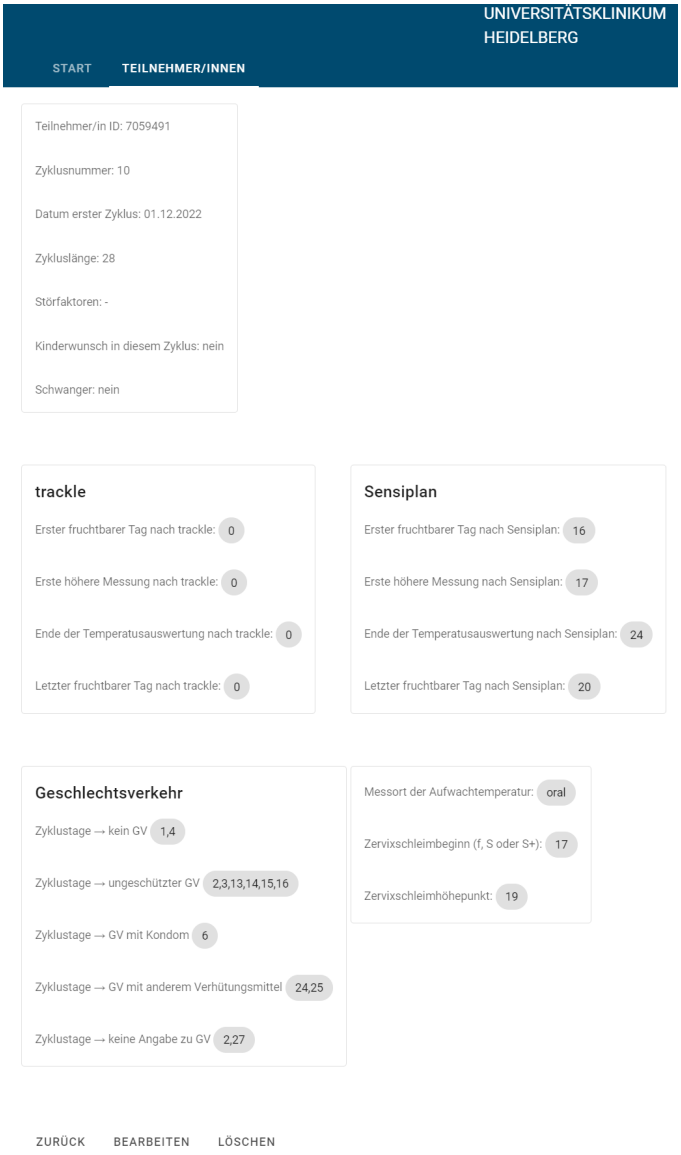

Figure 24: Single cycle page

The editing of a selected cycle is limited by the return of the back-end. Invalid intercourse, Sensiplan© or period tracker data can be edited or removed and replaced by a zero. To ensure consistency and correctness of the cycle data, the cycle length cannot be changed.

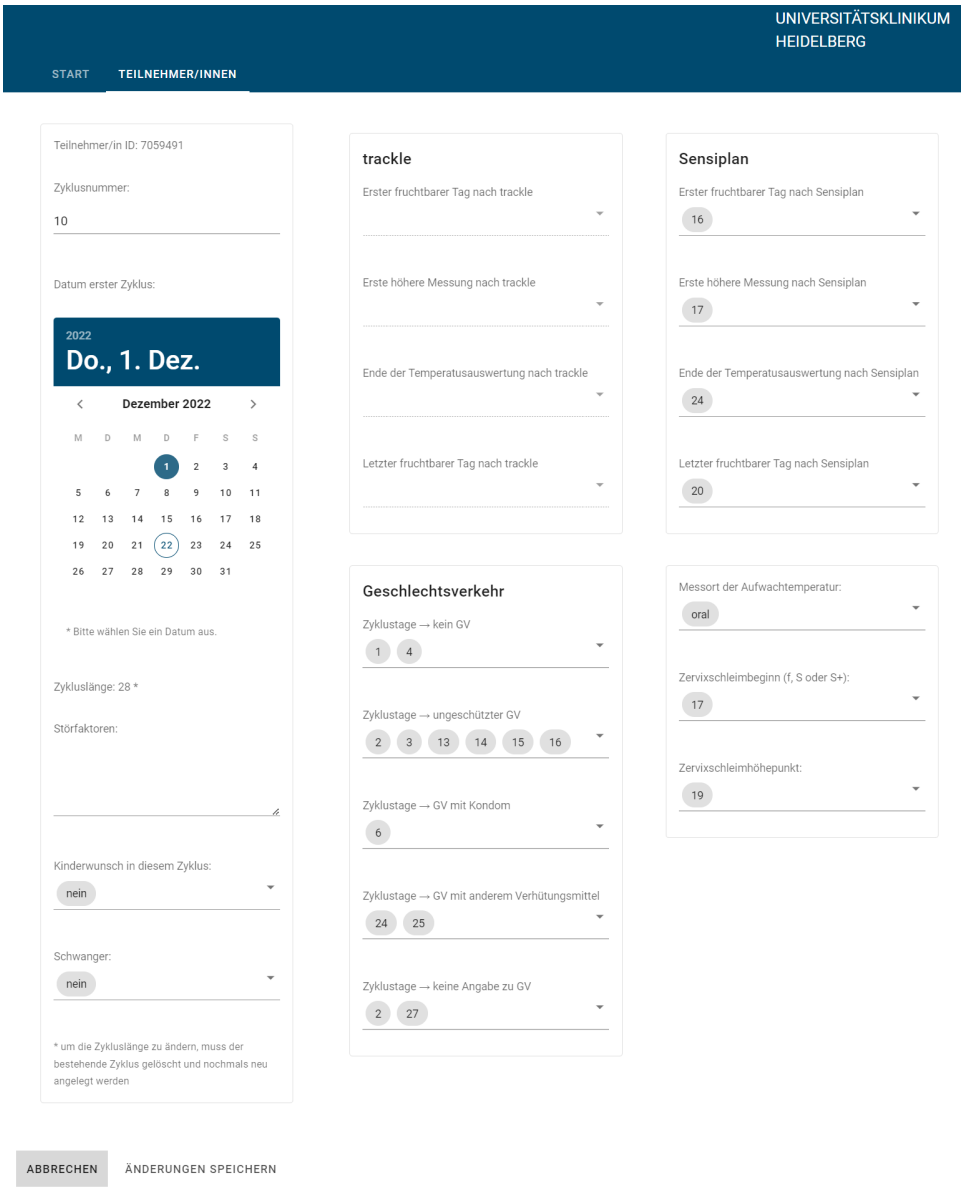

Figure 25: Edit cycle, *period tracker disabled*

5 Discussion

The final result of the project is a unique cross platform web application that supports study nurses and researchers in managing homogeneous data from a research back-end. Specifically, the application manages period data from both Sensiplan© and digital period trackers.

Vuetify was chosen as the graphical framework, with Vue as JS framework and Axios as a request library already in mind. To achieve an intuitive front-end, Vuetify mockups were created and realised with common features like a navbar and searchable tables. With Vues lightweight and flexible design, the web application is able to run on different devices. The back-end is mainly implemented with Axios requests, resulting in a simplified and easy-to-understand back-end. Axios can be equally lightweight and powerful. Both front-end and back-end have been implemented through Docker, which allows for simple deployment. As a persistent interface a FHIR server with the the FHIR API is used. Despite being very complex, the FHIR standard is well known in the medical field and facilitates syntactically improved and interoperable storage of data.

The primary back-end requirements were met by using the FHIR standard, which results in a persistent layer for storing data of participants and their respective period data. The chosen front-end technologies fulfill their purpose of providing a simple way to add, access, update and delete period tracker data for any given participant. By combining the back-end and front-end technologies all of the defined requirements have been met and fulfilled.

Most limitations of the web application correlated with the FHIR public test-server. Due to the large quantity of data, processing the entire database was not an option. Additionally, the public test-server contains data not matching the structure implemented in the back-end, which could potentially crash the web application. In addition, data entries were deleted sporadically.

However, just using a different FHIR test server would not solve these problems, since there is no publicly available FHIR test server which has no data collected by others. To solve this problem one would need to setup a personal FHIR server for this use case.

Other technological problems encountered during development included linking the *Patient* and *QuestionnaireResponse* resources, connecting the front-end and back-end, and the fact that participants who only took part in the new study were difficult to treat as data entries.

After implementing and combing all frameworks a web-application with an Axios back-end and the use of a FHIR API as a persistent data layer was created. This enables the creation, edition, update and deletion of participants and their respective cycle data. Additionally, it can be used with two different cycle tracker methods / apps, in this case Sensiplan© and Trackle. Moreover, data which is inserted will be validated by the front-end to ensure a high data quality in the database. Data consistency is achieved by tools like date-pickers instead of allowing users to freely enter data. To improve user workflow the front-end follows a simple colour-scheme and was implemented as a single-page application.

Complications occurred during implementation because Vuetify components returned certain data types that did not consistently match the existing data types in the back-end. This led to significant testing of the front- and back-end connection to ensure that the transferred data types matched the established types in the *QuestionnaireResponses* schemata. The pregnancy selection, for example, was implemented as a selection of Strings in the front-end, but stored as Booleans in the *QuestionnaireResponse*. This resulted in the string values being replaced with matching boolean values immediately before saving. The Boolean values returned by the back-end are in turn converted to string values while a HTTP-GET request is executed.

An issue directly related to the FHIR test server was the lack of examples on how to modify existing resources. Although HL7 provides good documentation, trouble-shooting errors was difficult without concrete implementations to compare to. This was most prominent during developing patch methods to gain access to nested data. On attempting a HTTP-PATCH the status code 200 was returned to the front-end while the respective information did not change in the back-end. No fix for the HTTP-PATCH method was found, hence the methods employing HTTP-PATCH methods were rewritten to use HTTP-PUT methods instead. This is less efficient as HTTP-PUT replaces entire objects opposed to HTTP-PATCH updating, creating and deleting objects as needed.

Another challenge was presented by the size of the FHIR server used to test the front-end. The amount of participants in the database prevented simply loading all present participants into the front-end. To bypass this, three *Patient* objects were hard-coded as participants. Those participants include all relative IDs and can be used to create different cycles with.

A further difficulty was to implemented a method in the back-end to display cycle data with a given period tracker or Sensiplan© ID, which turned out

to be very complex, as Sensiplan© and period tracker are independent data entries in the back-end database.

One of the issues that emerged during development was cycle length and data dependent on cycle length. While creating a cycle, its length is used to calculate and create the days in that cycle that are able to be selected and edited. Users could not edit specific days in a given cycle without setting the cycle length. By splitting the cycle creation dialogue into two pages, forcing users to set the cycle length on the first page, this was circumvented. When a cycle was edited, changing the cycle length given upon creation, data became inconsistent. Days not in range of the new cycle length were invalid and could not be processed, much like in an "array out of bounds" exception. This was solved making the cycle length immutable and instead forcing the user to recreate a cycle once a change in cycle length is detected.

A major weakness of the web application as of now is the lack of security, as there are no login mechanisms currently in place. This could be implemented relatively easily, and since no login method is provided, users could use their existing authentication system with the web application.

Furthermore the web application presently only supports Trackle as a digital period tracker and would have to be updated to work with additional digital period trackers. With the standards used in the project any period tracker should be able to be integrated using the same schemata as Trackle.

A further concern was empty values being passed from the front-end to the back-end. Incomplete inputs, especially empty inputs had to be saved as a dash in the back-end database, since an database post request should never be empty.

The interface implemented the FHIR standards, *Patient* and *QuestionnaireResponse* resources and enabled the separation of Sensiplan© and the period tracker *QuestionnaireResponse*. The ability for the two FHIR resources *Patient* and *QuestionnaireResponse* to work separately is a major benefit of our back-end implementation. While there are many FHIR-based web applications for research purposes, no projects dealing with period trackers were found during research.

Concluding by comparing the requirements and the results, including the limitations, it can be said that the overall purpose of a unique, cross-platform web application for managing, manipulating and storing data in an interoperable way has been achieved.

6 Conclusions

This project shows how a web application can be developed that supports study nurses and researchers in managing back-end data with a simple UI front-end. Some challenges for further development of this prototype are the lack of validation in the back-end and the non-generic use of the *QuestionnaireResponse* resource. This non-generic behaviour results from the dependence on the UKHD, Sensiplan©, and the period tracker. The paper and project can be used for future studies, or alternatively to build similar projects working with FHIR. Another option would be using the web application as a foundation to integrate a visualization dashboard to compare different PROs or even configure the web application to work with future studies that employ FHIR standards. To quantify the effectiveness of the web application compared to traditional data management a study would have to be done. Although it is safe to assume that like in virtually every comparable environment and increasing digitization in the medical world, a well thought out digital solution beats traditional methods.

7 References

References

[1] lucaskulla. *PMI*. original-date: 2023-01-10T18:55:37Z. Jan. 10, 2023. URL: https://github.com/lucaskulla/WebAppForStudyManagement (visited on 01/11/2023).

[2] *Overview - FHIR v4.3.0*. URL: https://www.hl7.org/fhir/overview.html (visited on 12/03/2022).

[3] *Safety - Sensiplan, the method for natural family planning*. Sensiplan®, ein Angebot der MW Malteser Werke gGmbH. URL: https://www.sensiplan.de/en/what-is-sensiplan/safety (visited on 12/03/2022).

[4] *Home - Sensiplan, the method for natural family planning*. Sensiplan®, ein Angebot der MW Malteser Werke gGmbH. URL: https://www.sensiplan.de/en (visited on 01/08/2023).

[5] *Getting Started | Axios Docs*. URL: https://axios-http.com/docs/intro (visited on 12/03/2022).

[6] *The Industry-Leading Container Runtime - Docker*. Sept. 28, 2021. URL: https://www.docker.com/products/container-runtime/ (visited on 12/05/2022).

[7] *Express - Node.js web application framework*. URL: https://expressjs.com/ (visited on 12/03/2022).

[8] Node.js. *About*. Node.js. URL: https://nodejs.org/en/about/ (visited on 12/03/2022).

[9] *Introduction | Vue.js*. URL: https://vuejs.org/guide/introduction.html (visited on 12/03/2022).

[10] *Material Design*. Material Design. URL: https://m3.material.io/ (visited on 12/03/2022).

[11] *Why you should be using Vuetify*. Vuetify. URL: https://vuetifyjs.com/en/introduction/why-vuetify/ (visited on 12/03/2022).

[12] *Overview | Vue CLI*. URL: https://cli.vuejs.org/guide/ (visited on 12/03/2022).

[13] *What is REST*. REST API Tutorial. URL: https://restfulapi.net/ (visited on 12/03/2022).

[14] *Swagger UI*. URL: https://hapi.fhir.org/baseR4/swagger-ui/ (visited on 01/08/2023).

[15] *Patient - FHIR v5.0.0-cibuild*. URL: https://build.fhir.org/patient.html (visited on 01/09/2023).

[16] *QuestionnaireResponse - FHIR v5.0.0-cibuild*. URL: https://build.fhir.org/questionnaireresponse.html (visited on 01/09/2023).

[17] *trackle Verhütungscomputer für hormonfreie Verhütung*. trackle GmbH. URL: https://trackle.de/ (visited on 01/08/2023).

[18] *Uniklinik Köln - Startseite*. URL: https://www.uk-koeln.de/ (visited on 01/08/2023).