FIRECARD

Firecard: A Medical Cloud Calendar

Based on MIDATA and FHIR, with Right to Privacy

Bachelor Thesis

Degree programme:	TI | Medical Informatics
Author:	Fahrni Alex
Thesis advisor:	Bignens Serge; Benoist Emmanuel
Project partner:	Von Kaenel François
Expert:	Van der Kleij Han
Date:	16.06.2022

Bern University
of Applied Sciences

Department
Division

Management Summary

Nowadays, medical appointments dates and hours are written on small cards and useful information is transmitted orally. However, cards can be lost and this information forgotten. If patients choose to write it all down in their electronic agenda, the issue of privacy arises. Indeed, these calendars are synchronized on servers abroad and often shared. So, the patients are no longer controlling their data. Therefore, a privacy preserving medical appointment system is important. This project shows the necessary technical basis and demonstrates the feasibility of doing so.

A literature search was carried to determine whether such projects have been done before. This was followed by research into the semantics of FHIR and the study of possible data flows. This then allows the classification of data according to their sensitivity from a confidentiality perspective. A discussion with a member of MIDATA, as well as with application developers from the BFH‘s I4MI Institute, also provided useful information for the orientation of the project. From this body of knowledge, the architecture of the project has been made, as well as the choices in terms of confidentiality.

No literature about projects using FHIR as an appointment manager was found. Sensitive data for a medical appointment are: title, reasons, location, descriptions, instructions, participants. All these data are stored securely and encrypted on MIDATA, and can be read on Firecard, the developed mobile application. The appointments can be exported to a usual calendar, but without the sensitive data, thus just the date and time.

The developed Firecard application demonstrates that it is possible to create a secure and privacy-friendly medical appointment system, based on a secure FHIR cloud. And this, without sacrificing the convenience of a usual calendar. FHIR shows its flexibility and power to be used in such applications, even if some shortcomings (e.g., missing owner field) exist due to the youth of the standard. Firecard's biggest problem is the Swiss healthcare. Currently, it is unthinkable that such an application could be released and used, due to the lack of incentives and the still underdeveloped digitalization. But I am confident that the project will be taken up and that the knowledge of this project will be used in practice in the future because it addresses an issue that will arise in the years to come.

Contents

1 Introduction

1.1 Initial Situation

Worldwide, the no-show rate for ambulatory medical appointments is between 15% and 30% [1]. This leads to a reduction in the availability of appointments and an increase in healthcare costs [1]. Several solutions exist to reduce the no-show rate, such as SMS, phone calls, mail, or email. Institutions use these systems at their level. However, there is no generic ready-made solution to be included in mobile health applications, which not only provide reminders but also create and manage medical appointments. Indeed, if a clinic wants to develop an application that includes a reminder and invitation function, they will have to use a system that is already on the market. A system that is not necessarily designed to handle medical data in the first place, and which does not pay respect to privacy potentially. Another solution is to create an in-house and proprietary system.

Another problem is the management of appointment scheduling. In general, the patient will find a suitable date with the medical secretary and will receive a small card containing the date and time. Important information such as pre-appointment instructions, location or with whom the patient has an appointment will not be specified on the card and will be transmitted verbally. This leaves it to the patient to take care of writing it all down. Something frequently done by patients is to write everything down in their electronic calendar on their smartphone, but this also raises privacy issues. Indeed, these calendars are usually synchronized on clouds that do not respect privacy, or with other people, such as work colleagues for example. This is something that should be avoided because depending on the address, the name of the doctor or the instructions, it can tell a lot about the person's health. It is therefore essential to secure this information.

The Firecard project was created based on this original premise. The aim was to imagine and demonstrate the feasibility of a mobile and reusable system that would take care of calendar functions. This proof of concept is in the form of a mobile application called Firecard, and lays the foundations for a doctor to patient and patient to patient appointment management system, as well as a reminder system. The watchword during all phases of the project was “privacy”. To enable the exchange of data between the different actors, it is essential to have a secure cloud that respects the privacy of the users. It is also necessary to use recent and reliable technology that has been created for the medical field. Naturally, the cloud platform of the MIDATA cooperative was chosen. In addition to its long-standing collaboration with the Bern University of Applied Sciences. It offers a FHIR server and a secure communication system with it. Furthermore, the patient data stored on MIDATA remains their property and the cooperative is committed to respecting the privacy of its users.

1.2 Research Questions

Starting from this, i.e., the use of FHIR and MIDATA, several research questions were defined to guide the project. It is important to define the context of use of the project, as well as the target. The first research question is therefore:

- Which actors and scenarios are involved in mobile application projects regarding calendar functions?

FHIR is composed of dozens of resources and offers many possibilities, among which choices must be made that are adapted to the project. This is a crucial step because FHIR directly defines the semantics of the elements used by the project. The second question the project must answer is:

- How should a FHIR-based mobile medical calendar be built?
  - Which FHIR resources are involved?
  - Which information are mandatory, and which are optional?

MIDATA is not just a simple FHIR server, it is a platform offering different services around the server. So, we must look at all the possibilities and choose what is best for Firecard. The third research question is:

- How should an application that uses MIDATA as a cloud service be built?
  - What are the solutions and the possibilities proposed by MIDATA?
  - What are the limitations of MIDATA?
  - How must the communication to MIDATA be designed?

Once these questions are answered, all the keys are here to conceptualize and develop the Firecard application. With this result, the feasibility and the demonstration of the overall concept is done. The research question is the following:

- What should the software architecture of a mobile medical calendar module based on FHIR and MIDATA look like?

2 Methodology

2.1 Literature Research

The literature search was conducted using three tools: PubMed and Google Scholar for the databases of scientific articles and ResearchRabbit to study the metrics of the articles found, but also to find similar ones, and to see the publications citing those found. This research had several aims. The first was to find articles demonstrating the importance of medical appointment reminders. The second goal was to find articles that demonstrated the importance of confidentiality in the processing and storage of medical data, and the related security concerns. The third one was to find out if such projects were already been made using FHIR. For the latter, an email was sent to Dr. Marco Engeler, president of the VSFM, as well as to the IT director of a cantonal hospital, who however did not follow up on it. Different keywords were used to achieve these goals: (appointment reminder systems), (medical data privacy), (FHIR appointment).

2.2 Requirements

After the literature search, a project charter was created and submitted for validation by the project supervisors. After that, the project specifications and research questions were defined and submitted for validation. You can find these documents in appendix 1 and 2.

2.3 Management

The project has 5 phases: initiation, planning, implementation, monitoring and controlling, closure. Six milestones have been defined to define the progress of the project and to have a common thread. The initiation and planning phase lasted the first 3 weeks and the aim was to reach milestone 1: definition of the proof of concept to be delivered. The execution phase followed the two previous ones and lasted 11 weeks. Scrum was used for this one, although the project was done alone. It was divided into four sprints, the goal of each sprint being to reach a new milestone. The last one being: release candidate reached. The monitoring and controlling phase lasted throughout the project and consisted of weekly meetings with the two project supervisors: Benoist Emmanuel and Bignens Serge. But also in two meetings with the expert Han van der Kleij. These meetings served to inform about the progress of the project, to take stock of the situation, and to discuss and clarify the work ahead. Meetings with a MIDATA developer Alexander Kreutz, as well as two developers from the I4MI institute, von Kaenel François and Hess Gabriel, were also part of this phase. The purpose of these meetings was to discuss the technical aspects of the project. The closure phase also lasted throughout the project and consisted of writing and creating all the project deliverables. More information on the project management can be found in appendix 3.

2.4 Implementation

Implementation of Firecard was done with the cross-platform framework React-Native to ensure a wide availability on both iOS and Android. Although to date only the Android version is available and has been tested due to time constraints. But the iOS version will only take a short time to release, as more than 95% of the code is common. The choice of React-Native was made because the interested party in the project, I4MI, uses this technology to develop their applications. Moreover, React-Native resources could be provided by them in order to save time. As the project is a proof of concept, the Firecard application will not be made available on the stores. The current standards for React-Native and the communication protocols used by MIDATA were used. For more details about the versions and libraries used, please refer to Appendix 4.

2.5 Testing

Unit tests have been written to test the different parts of the Firecard code that can be tested in this way. The testing of the more global components and functionalities could not be tested automatically in this project and were therefore tested manually. Usability tests were carried out with real patients and relatives. The usability test was divided in three phases: an interview asking testers for their name, age, level with smartphone and how do they deal with the Firecard issue now. In the second phase, the patients have access to Firecard and must do a set of actions. The last phase in still an interview, but focus on the experiences they have had, and if they have recommendations.

3 Results

3.1 Results from Literature Research

Several articles about appointment reminders could be found, including [1]–[3]. On the other hand, nothing has been found on systems like Firecard in idea. Even more globally, very little information is available on FHIR used as an appointment manager and not as an exchange format. Dr. Marco Engeler, president of the VSFM, responded that this is quite normal, as appointment management is a complex field that has already developed over 20 years ago. The players in this field today are not ready to take the step to change everything for FHIR. However, he says that it would be technically possible.

3.2 Actors and Scenarios involved in Mobile Health Applications regarding Calendar Functions

There are two main scenarios. In the first, a doctor sends a series of invitations to his patient, who can accept or decline them. In the second, it is patient-to-patient invitations, which is useful for patient groups or associations, for example. Surely, for perfectly healthy people, using the Firecard application to go to the dentist or doctor only once a year makes little sense. That is why it's an application that would be more suitable for patients with chronic diseases, who need a real follow-up between their doctor and themselves. They are also more likely to participate in support groups.

3.3 Build a FHIR based Mobile Medical Calendar

This section and the Firecard project is based on FHIR R4.

To build a FHIR based mobile health calendar, we must define which resources could be involved in this theme. FHIR is divided in five categories [4]. For this project, it is the workflow categories that provide everything we need. The aim of this module is to coordinate activities in a system or between different ones. Five resources are interesting for Firecard:

- Task
- Appointment
- AppointmentResponse
- Schedule
- Slot

At the start of the project, it was discussed to also provide a reminder system such as a medication intake. The **Task** resource is suited for this kind of use. It defines something to do at a certain moment. But due to time limitation, this feature was abandoned.

The **Appointment** resource is the main component of this project. It defines the semantic of Firecard. Patient or physician will create this resource, and it contains all information about the appointment.

AppointmentResponse is used to answer to appointment request across different system. Here we just have one, MIDATA, so this resource is not needed.

Schedule is an interesting resource. It represents the availability of a related resource. That means, with a schedule, we can know if a person or a device is available for a defined time slot. It could be interesting for future version, where patients could be able to see the availability of their doctors and asking for an Appointment in an empty slot. Currently, Firecard do not use this resource because patients can notsend appointment to doctors.

Slot is a resource representing a starting and an ending date and time. It is used in Schedule or Appointment resource. But Firecard do not use it because it does not provide anything as it stands.

As said before, Firecard only use the **Appointment** resource. Here are the fields used by Firecard:

- **status**: It is mandatory; it represents the current state of an appointment. Firecard sets this value with “proposed” when an Appointment is created. When data is fetched, appointments with “cancelled” are excluded. At the beginning, “cancelled” appointments were shown because it is interesting to know that an appointment has been cancelled. But on MIDATA, we cannot delete a resource. So, the only means to do that was to use this field.
- **description**: It is not mandatory; it represents the title of an appointment.
- **start**: It is not a mandatory field, but in Firecard it will always be set. This field represents the starting date and time of an appointment.
- **end**: It is not mandatory, but in Firecard it will always be set. This field represents the ending date and time of an appointment. Firecard pay attention that this field is always greater or equal to **start**.
- **minutesDuration**: It is not mandatory, but in Firecard it will always be set. This field is the absolute difference between **start** and **end**.
- **comment**: It is not mandatory; in Firecard it represents the description of an appointment like the description in an iCal event.
- **patientInstruction**: It is not mandatory; This field is displayed on Firecard only if it is set. Patients cannot set this field when there are creating an appointment. So, only physicians can set it. It represents a set of instructions that the patient should follow before an appointment, like “not to eat”.
- **participant**: it is mandatory; it is a list containing participants; a participant could be a patient, a physician, but also a room or a location. In Firecard, **Practitioner**, **Patient, and Location** are participants.

The Location of an appointment is only valid in its context in Firecard. That is why this resource will be stored in the contained field of an appointment. Because it does not make sense outside this appointment.

It is important to determine which data is sensitive and which are not. The diagram below shows that:

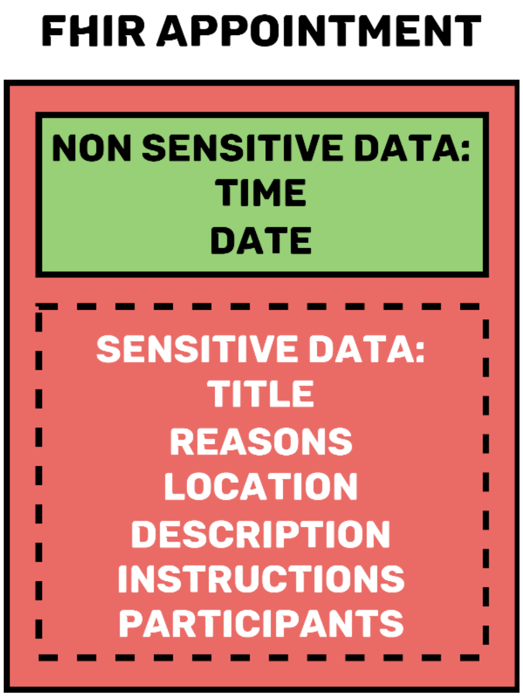


Figure 1: Used FHIR Appointment fields separated by their sensitivity.

3.4 Build an Application using the MIDATA Cloud

MIDATA offers several solutions to connect a software to its platform: A user application, a plugin, an import service, a backend service, an external service, and a project analyser.

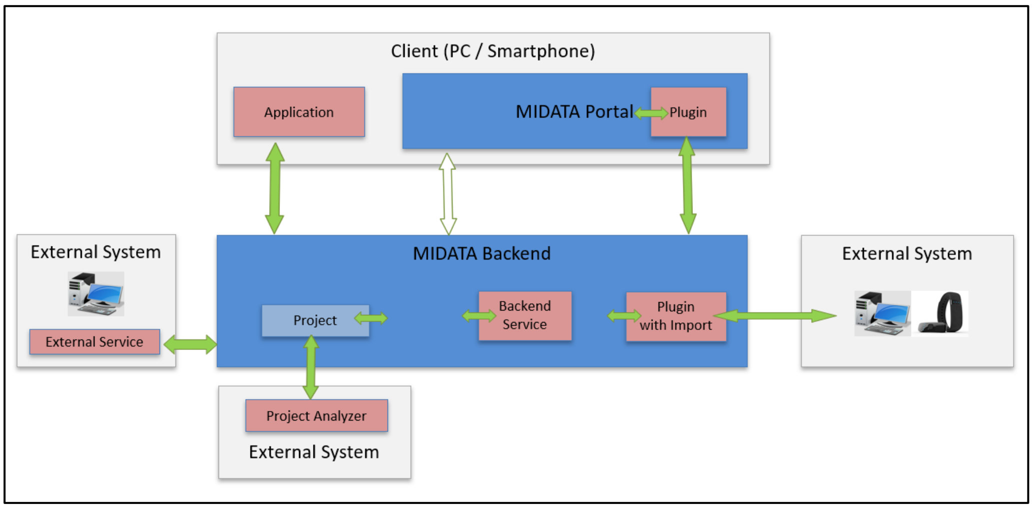


Figure 2: Types of software that may be connected to MIDATA. [Source: MIDATA]

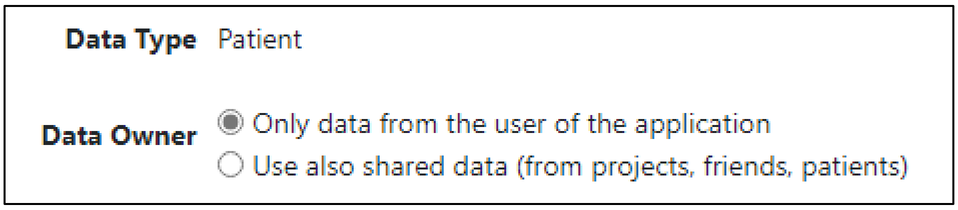
As we can see on Figure 1, MIDATA provides two elements: a portal and the backend. The portal is used to create an account, to view the data and to manga the account. The backend provides the FHIR server, but also an authentication and authorization server.

For Firecard, the Application solution was used. The authentication is handled by an OAuth2 process. After getting a developer account, an application must be created on the MIDATA dashboard. For Firecard four things were important during this process: the name and the redirect URL because these elements are being used for the authentication, and it must be specified that user must have an email on their account because Firecard use the emails as identifier, especially for invitations. The target role must be set on all roles because Account Holder (patient) and Healthcare Provider (Physicians) should be able to use the application.

```
host=https://test.midata.coop
client_id=firecard
redirect_URL=firecard.app:/oauthredirect
auth_endpoint=/authservice
token_endpoint=/v1/token
```

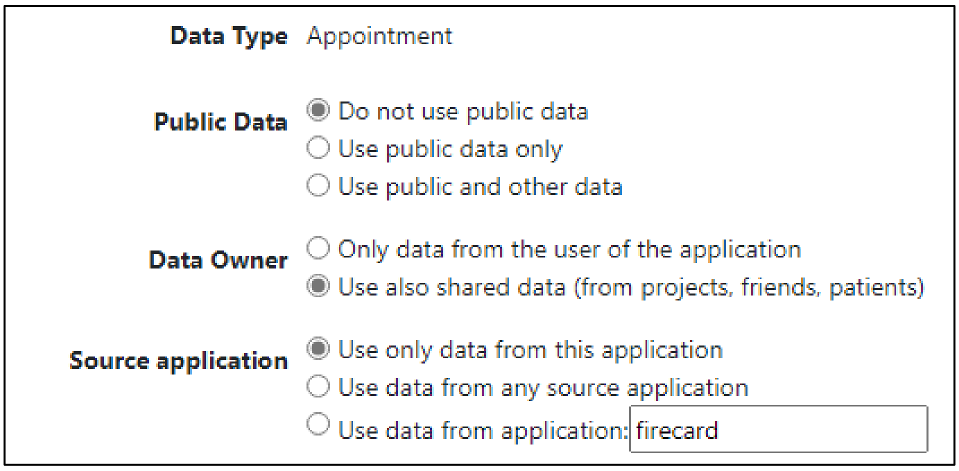
Figure 3: Configuration of Firecard for the OAuth2 authentication.
The client_id and redirect_url must be entered on the MIDATA dashboard.

On MIDATA, the access filters of the application must be specified. That means, which FHIR resources can be read and written by the application. For Firecard it was simple as we just need a read access for the Patient resource and a read and write access on Appointment.



Data Type	Patient
Data Owner	● Only data from the user of the application ○ Use also shared data (from projects, friends, patients)

Figure 4: Data access filter for Patient resource.



Data Type	Appointment
Public Data	[✓ Do not use public data] / Use public data only / Use public and other data
Data Owner	Only data from the user of the application / [✓ Use also shared data (from projects, friends, patients)]
Source application	[✓ Use only data from this application] / Use data from any source application / Use data from application: firecard

Figure 5: Data access filter for Appointment resource.

Some limitations or problems with MIDATA were encountered during the development of Firecard. Firstly, it was impossible to use the FHIR keyword ":not", it is used to exclude a code value when doing GET request on a FHIR server. In Firecard, it would have been used so when fetching appointments on MIDATA:

```
`/Appointment?date=eq${date}&_sort=date&status:not=cancelled`
```

Instead of:

```
`/Appointment?date=eq${date}&_sort=date&status=proposed,pending,booked,arrived,checked-
in,waitlist`
```

The second limitation was more a bug. In FHIR Appointment resource, there is no field to save the creator of a resource. However, MIDATA added this functionality with an extension. Firecard use this to know if the logged-in user is the creator of the displayed appointment. If it is the case, a delete button is shown. Because only the creator of a resource can delete a resource (as said in chapter 3.4, users cannot delete a resource, it just set the status to "cancelled").

Finally, the fact that users cannot delete resources is also a limitation. It is understandable that MIDATA does not allow it because it is better to have a traceability of what have been made.

Once the application was created on Firecard, the OAuth2 connection can be established. On figure below, can you find a simplified OAuth2 sequence diagram:

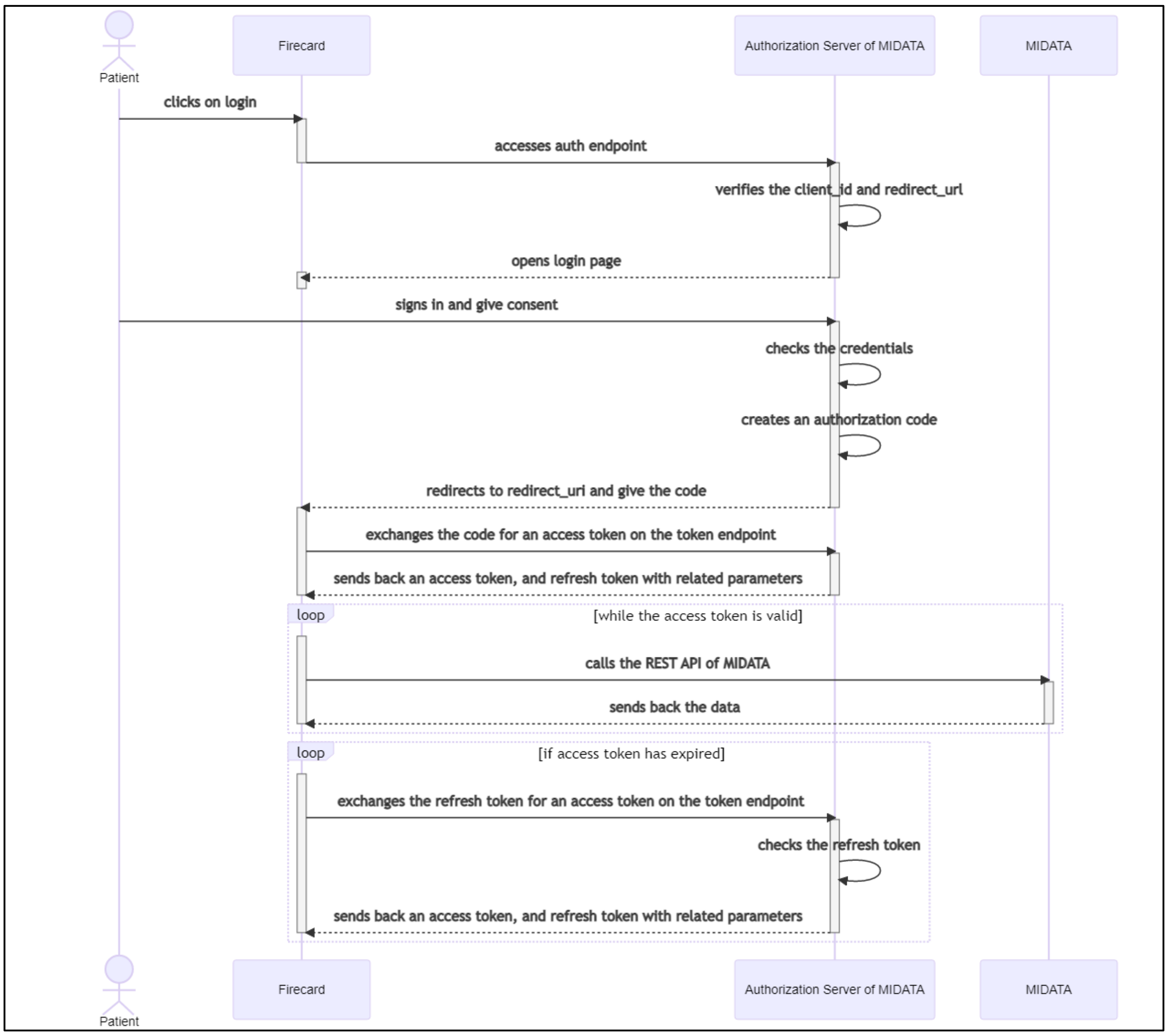
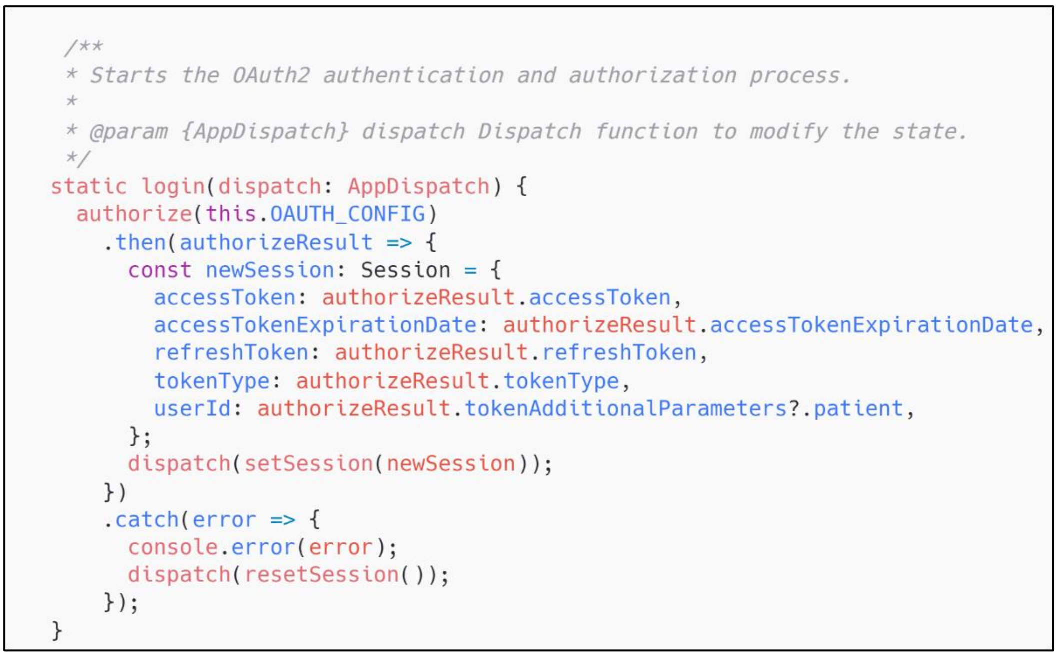

Patient
Firecard
Authorization Server of MIDATA
MIDATA
clicks on login
accesses auth endpoint
verifies the client_id and redirect_url
opens login page
signs in and give consent
checks the credentials
creates an authorization code
redirects to redirect_uri and give the code
exchanges the code for an access token on the token endpoint
sends back an access token, and refresh token with related parameters
loop
[while the access token is valid]
calls the REST API of MIDATA
sends back the data
loop
[if access token has expired]
exchanges the refresh token for an access token on the token endpoint
checks the refresh token
sends back an access token, and refresh token with related parameters
Patient
Firecard
Authorization Server of MIDATA
MIDATA

Figure 6: OAuth2 sequence diagram with Firecard and MIDATA

By chance, it exists a library called “react-native-app-auth” that handles all this process for us in React Native. A configuration object with data from Figure 3 must be supplied to the “authorize” method, and the result is stored in a “Session” object.



```
  /**
   * Starts the OAuth2 authentication and authorization process.
   *
   * @param {AppDispatch} dispatch Dispatch function to modify the state.
   */
  static login(dispatch: AppDispatch) {
    authorize(this.OAUTH_CONFIG)
      .then(authorizeResult => {
        const newSession: Session = {
          accessToken: authorizeResult.accessToken,
          accessTokenExpirationDate: authorizeResult.accessTokenExpirationDate,
          refreshToken: authorizeResult.refreshToken,
          tokenType: authorizeResult.tokenType,
          userId: authorizeResult.tokenAdditionalParameters?.patient,
        };
        dispatch(setSession(newSession));
      })
      .catch(error => {
        console.error(error);
        dispatch(resetSession());
      });
  }
```

Figure 7: Login method of Firecard using authorize method.

It is the same process to refreshing the token, but with the “refresh” method.

3.5 Software Architecture of Firecard

3.5.1 Architecture

The Firecard project implies many components. On the diagram below, you can see with which components interact the Firecard application. And, which data is transferred.

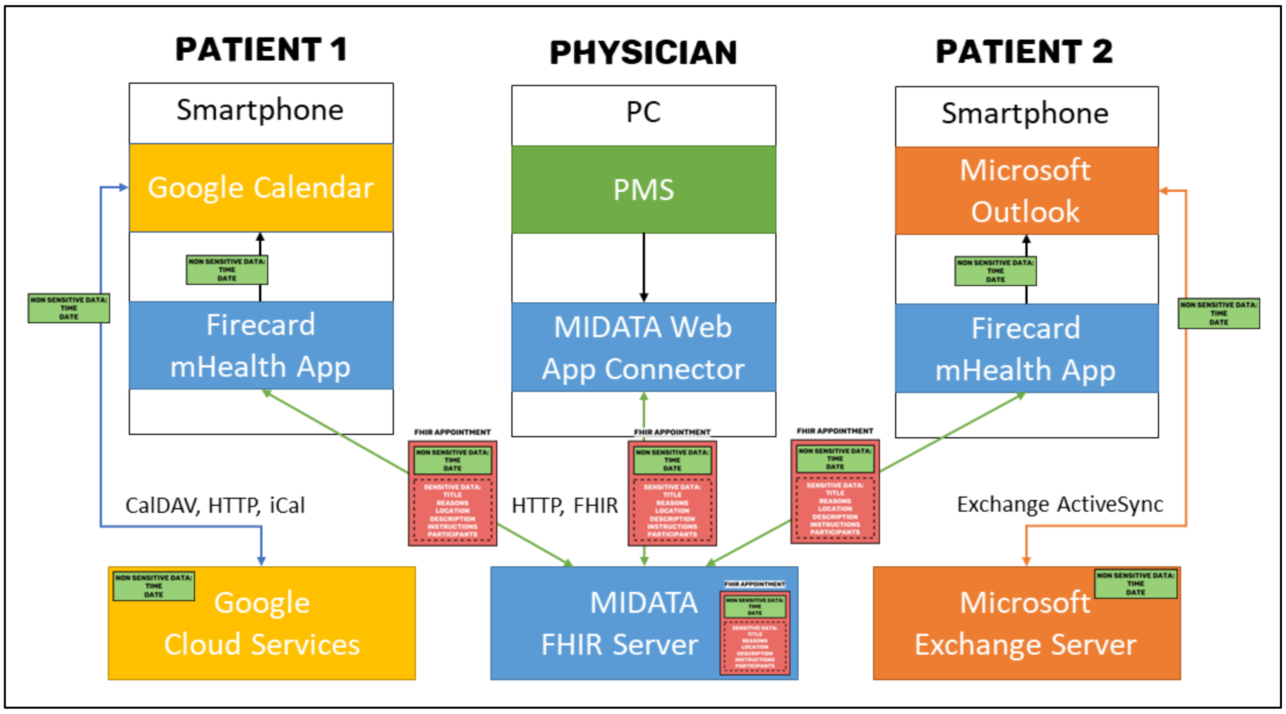


Figure 8: Architecture of the Firecard project.

Firecard uses a state manger, called Redux, to store the data that may be used across the different screens. A store is created, it contains all the saved data divided into smaller states.

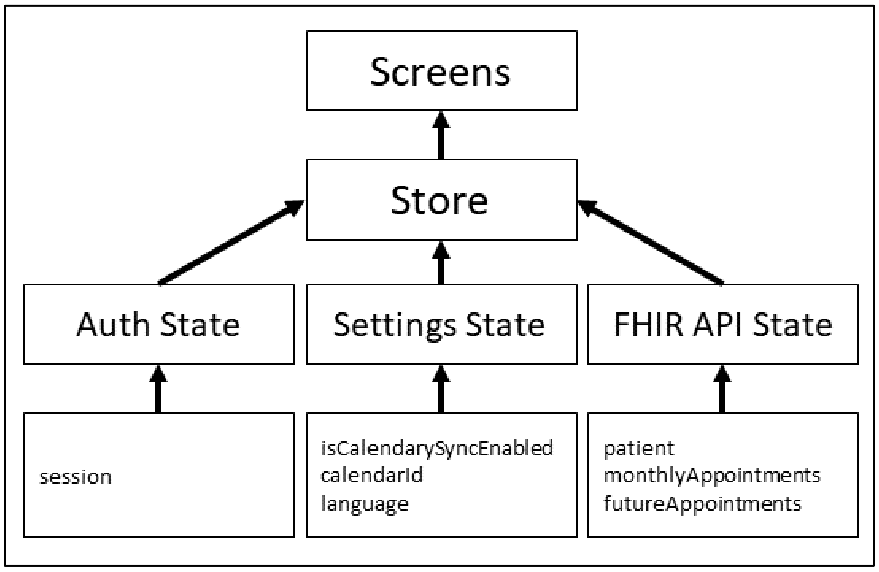


Figure 9: Redux store structure of Firecard.

To modify these data, we must dispatch the new data to a reducer function. Only reducers can change the state.

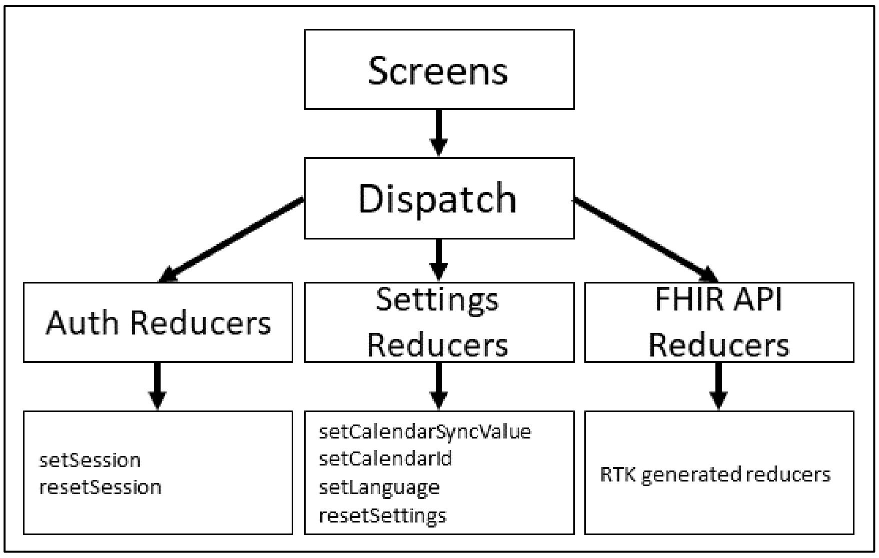


Figure 10: Redux reducers of Firecard.

The FHIR API State and Reducers use a technology called RTK. It provides powerful solutions for data fetching and mutating. It can also put the data in a cache to avoid unnecessary request.

We cannot save object coming from JavaScript classes in a Redux store. That is why Firecard do not use classes for the Session, the Patient, or the Appointment. These elements are types or interfaces instead. Different utility classes were written to perform operation on these elements.

The advantage of using Redux is that we can easily add a persistor, a technology that will save all data when the application is closed or put in background. In Firecard, the persistor saves the data in the Android SharedPreferences and in the iOS KeychainService. Attention, SharedPreferences should be encrypted on Android because they can be read by other applications. “redux-persist-sensitive-storage” package have to update its code to use a newer version of “react-native-sensitive-info”because current version do not support encryption. Many solutions were tried to add this encryption, but it did not work. Most of the time, the length of the value was limited to 2048 bytes, that is not sufficient for Firecard. After discussion with supervisors, it was decided not to address this concern in this version of the project. Solutions to this problem could be to use a local database like SQLite.

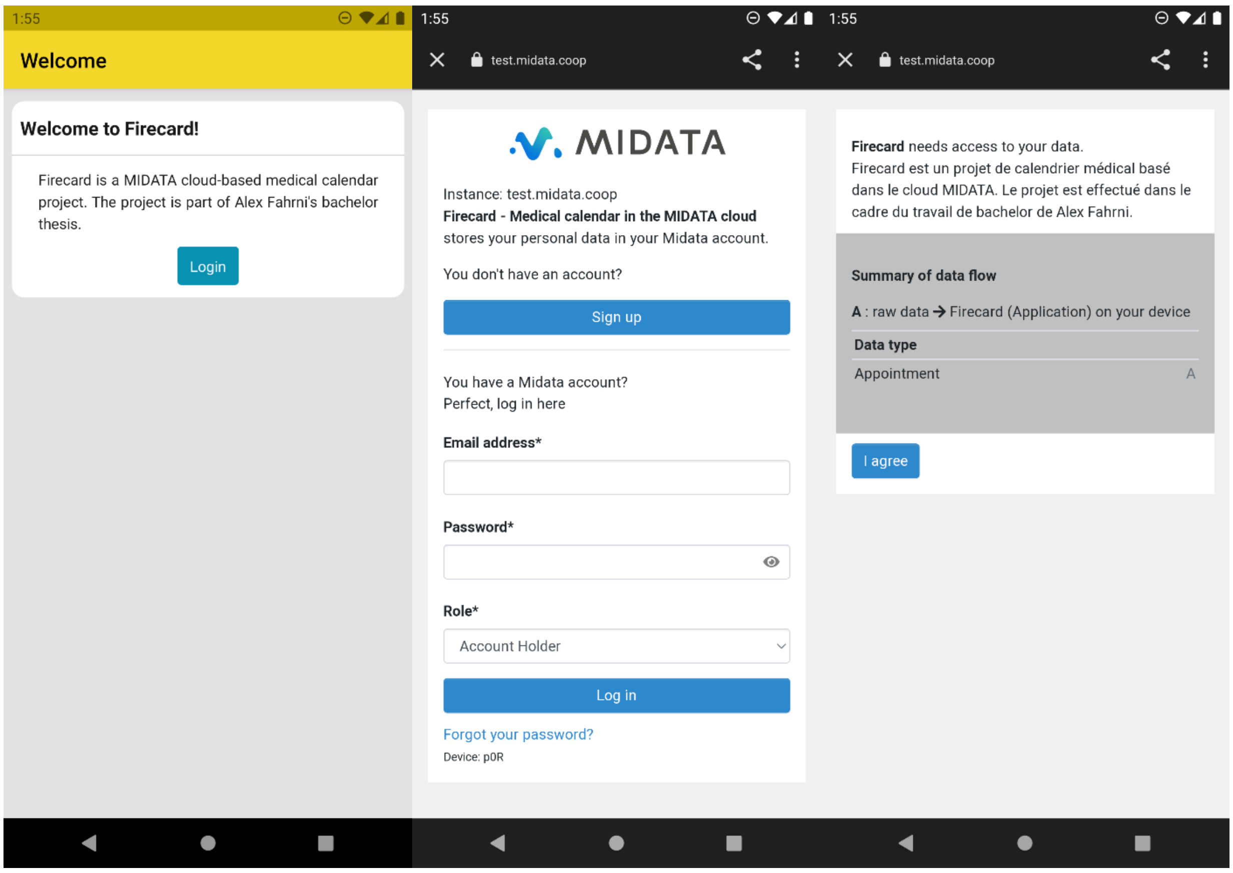
All written Firecard is stored under the “app” folder. Inside, files are organized in different categories according to their functions.

Table 1: Code structure of Firecard.

Directories	***For…***
app/components	custom developed components
app/containers	Screens
app/hooks	React Hooks
app/models	type, interface, and props definition
app/resources	static resources like fonts or pictures
app/store	all elements concerning the Redux store
app/utils	the utility classes

3.5.2 Screens

The Firecard application is composed of six screens. The first one is the first page the users will see. It displays some information about Firecard and has a login button. Clicking this button will open the web browser and will redirect to the MIDATA login page. If the patient never connected to Firecard before, a consent page with the used data will be shown.



1:55

Welcome

Welcome to Firecard!

Firecard is a MIDATA cloud-based medical calendar project. The project is part of Alex Fahrni's bachelor thesis.

Login

1:55

test.midata.coop

MIDATA

Instance: test.midata.coop

Firecard - Medical calendar in the MIDATA cloud stores your personal data in your Midata account.

You don't have an account?

Sign up

You have a Midata account?
Perfect, log in here

Email address*

Password*

Role*

Account Holder

Log in

Forgot your password?

Device: p0R

1:55

test.midata.coop

Firecard needs access to your data.
Firecard est un projet de calendrier médical basé dans le cloud MIDATA. Le projet est effectué dans le cadre du travail de bachelor de Alex Fahrni.

Summary of data flow

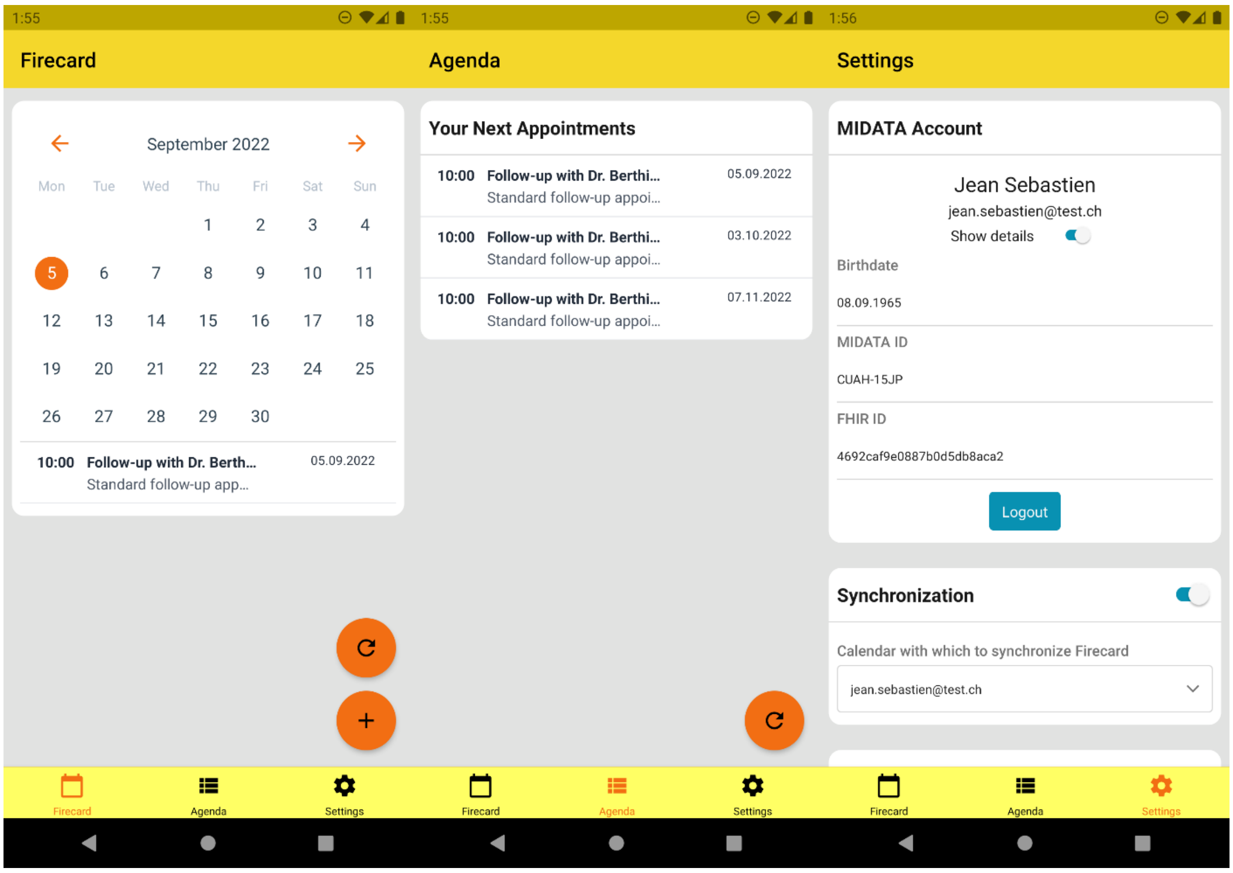
A : raw data → Firecard (Application) on your device

Data type	
Appointment	A

I agree

Figure 11: Login screen of Firecard, the login page of MIDATA and the consent agreement.

Once the patient logged in, he can navigate through 3 screens with a tab navigator. The default screen contains a calendar that shows the appointments for the selected day. Patient can navigate through the months and days. He can also refresh the appointments or create one. The second screen is the agenda. It provides a list of all future appointments, user can also refresh the appointments. The settings screen shows information about the logged user, a button to logout, a switch to activate the synchronization with regular calendar, which allows selecting a calendar installed on the device. There is also a language option, but only English is available yet.



Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

10:00 Follow-up with Dr. Berth... 05.09.2022
Standard follow-up app...

Your Next Appointments

10:00 Follow-up with Dr. Berthi... 05.09.2022
Standard follow-up appoi...

10:00 Follow-up with Dr. Berthi... 03.10.2022
Standard follow-up appoi...

10:00 Follow-up with Dr. Berthi... 07.11.2022
Standard follow-up appoi...

MIDATA Account

Jean Sebastien
jean.sebastien@test.ch
Show details

Birthdate
08.09.1965

MIDATA ID
CUAH-15JP

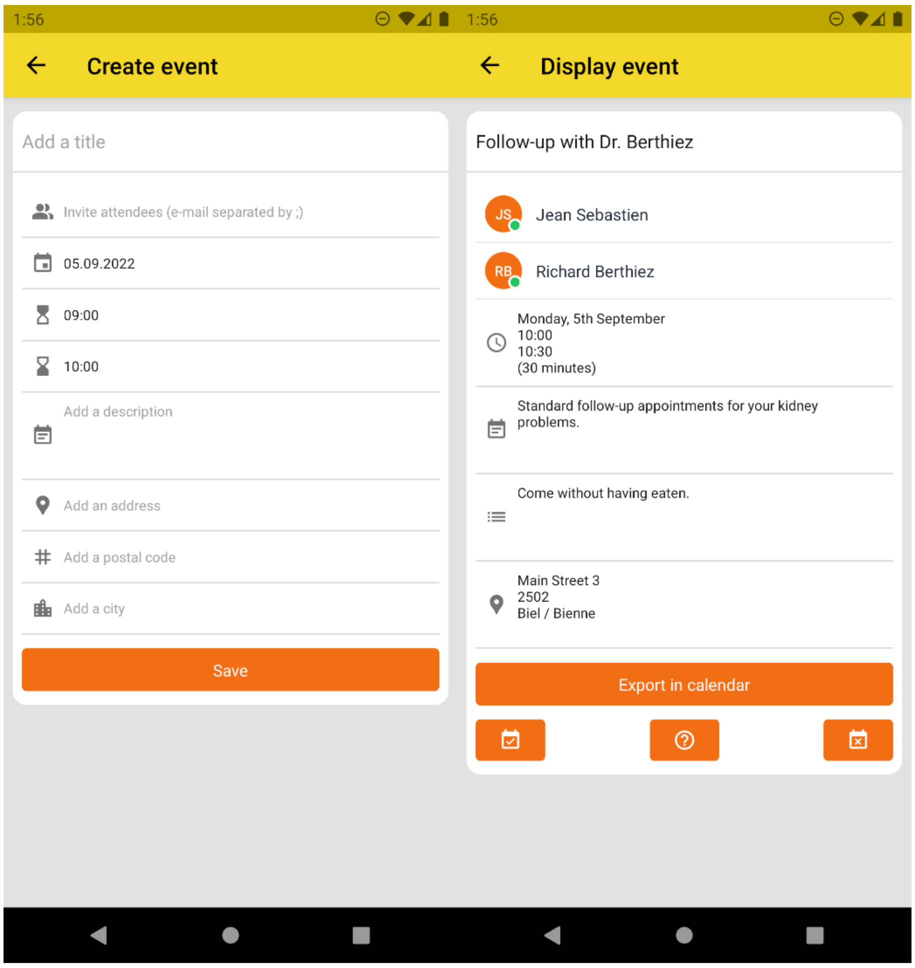
FHIR ID
4692caf9e0887b0d5db8aca2

Synchronization

Calendar with which to synchronize Firecard
jean.sebastien@test.ch

Figure 12: Default, agenda, and settings screens of Firecard.

The two last screens are the create and the display event screens. The first one allows a patient to create an appointment with other patients, for the use case of patient supports groups, for example. The display event screen shows a selected appointment. The patient can answer to the appointment (accept, tentative, decline). But also export it in a regular calendar without the sensitive data, or delete it if he is the owner.


1:56

Create event

Add a title

Invite attendees (e-mail separated by ;)

05.09.2022

09:00

10:00

Add a description

Add an address

Add a postal code

Add a city

Save

1:56

Display event

Follow-up with Dr. Berthiez

JS Jean Sebastien

RB Richard Berthiez

Monday, 5th September
10:00
10:30
(30 minutes)

Standard follow-up appointments for your kidney problems.

Come without having eaten.

Main Street 3
2502
Biel / Bienne

Export in calendar

Figure 13: Create and display event screens.

3.5.3 Dataflows

There are many dataflows involved in Firecard. In this chapter, you will see sequence diagrams representing them.

Firstly, there is a case that explain how healthcare professional can send appointment to MIDATA. That is the theory because there is currently no practice management system that communicate directly with MIDATA. And to simulate this, we have to access the MIDATA API Sandbox.

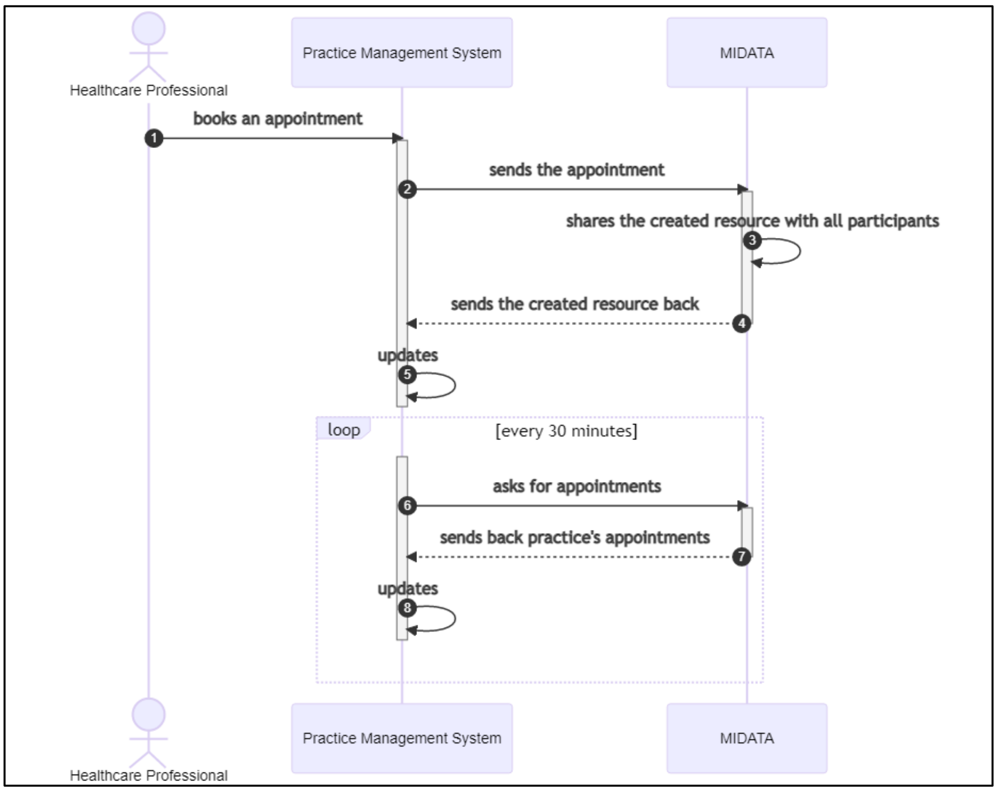

Healthcare Professional
Practice Management System
MIDATA
books an appointment
sends the appointment
shares the created resource with all participants
sends the created resource back
updates
loop
[every 30 minutes]
asks for appointments
sends back practice's appointments
updates
Healthcare Professional
Practice Management System
MIDATA

Figure 14: Healthcare professional books an appointment.

Then there is the case when a patient books an appointment with other patients. The part when it checks every 30 minutes and notifies the patient was not implemented.

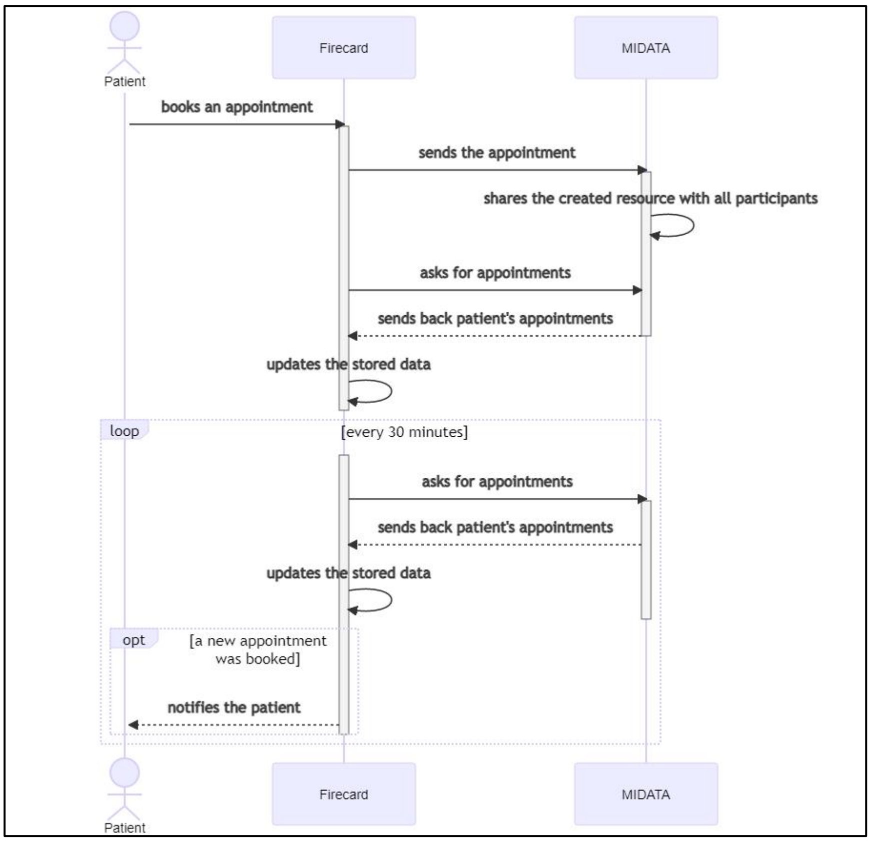


Figure 15: Patient books an appointment.

The patients can also respond to an appointment.

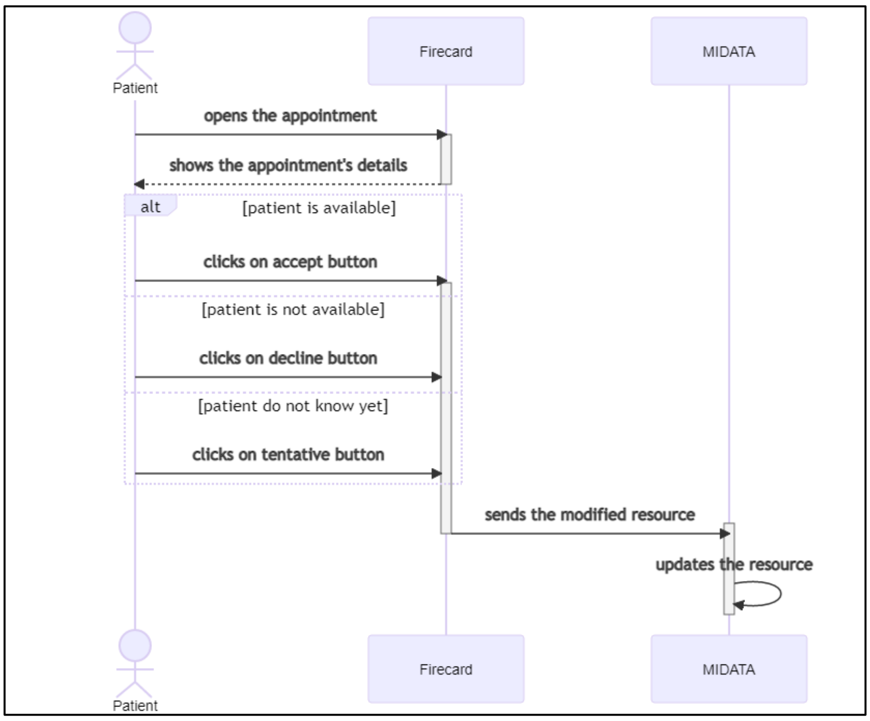


Figure 16: Patient responds to an appointment.

If a patient deletes an appointment, the status of this appointment will be set to “cancelled”. When Firecard fetches the appointments, it will exclude those with this value.

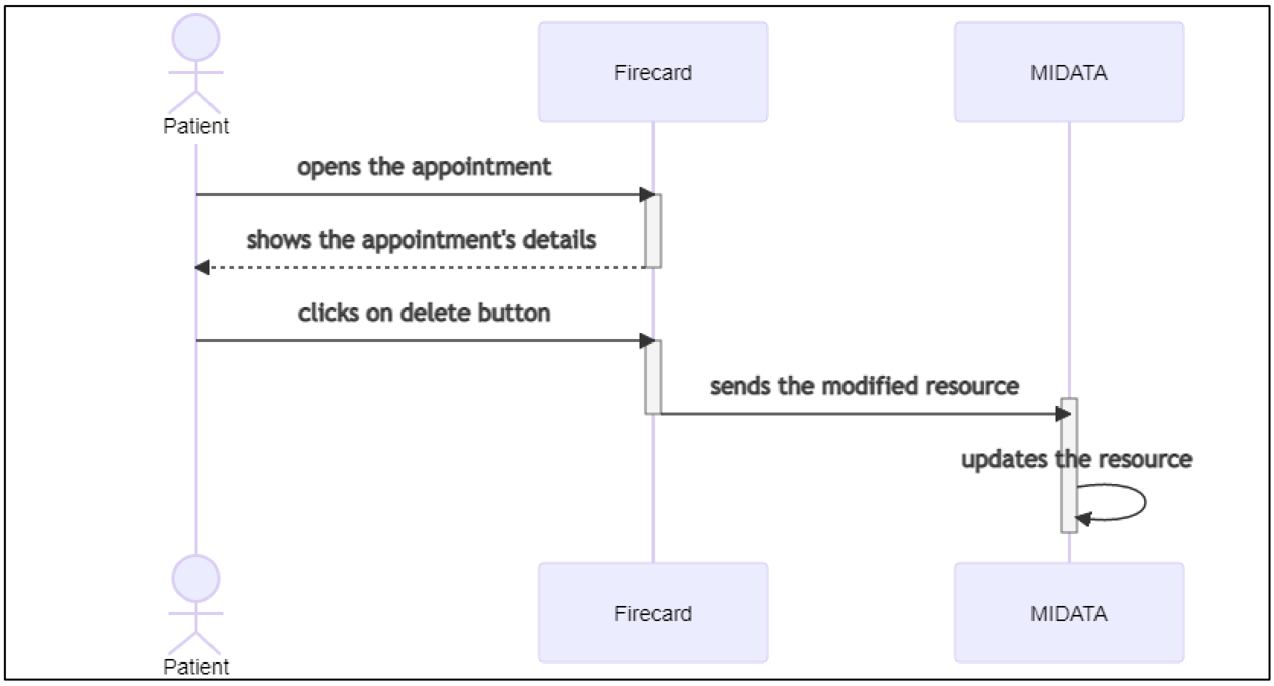


Figure 17: Patient deletes an appointment.

Patients can also export the appointments without sensitive data.

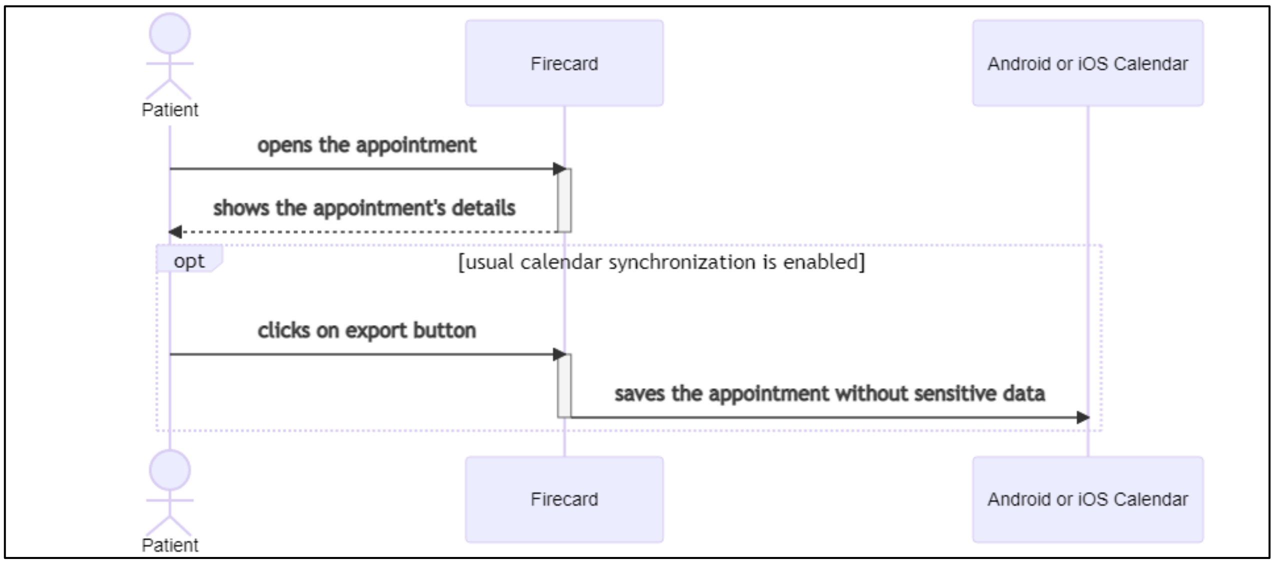
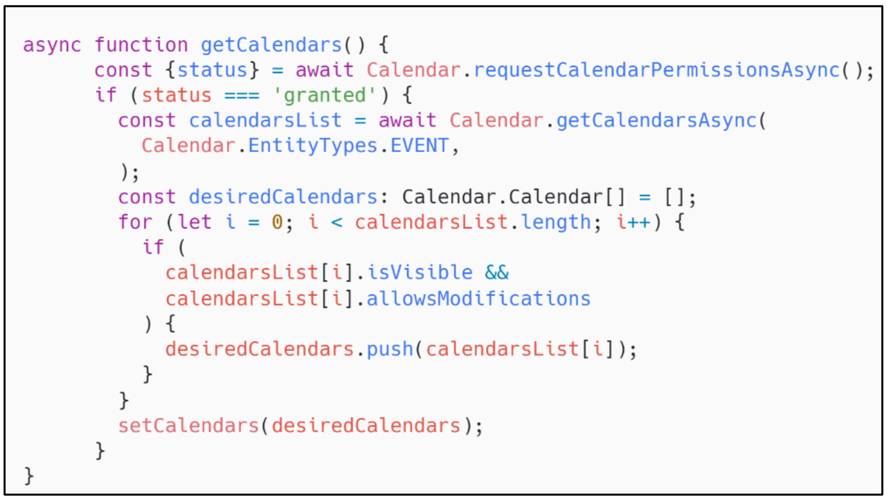


Figure 18: Patient exports an appointment.

3.5.4 Export

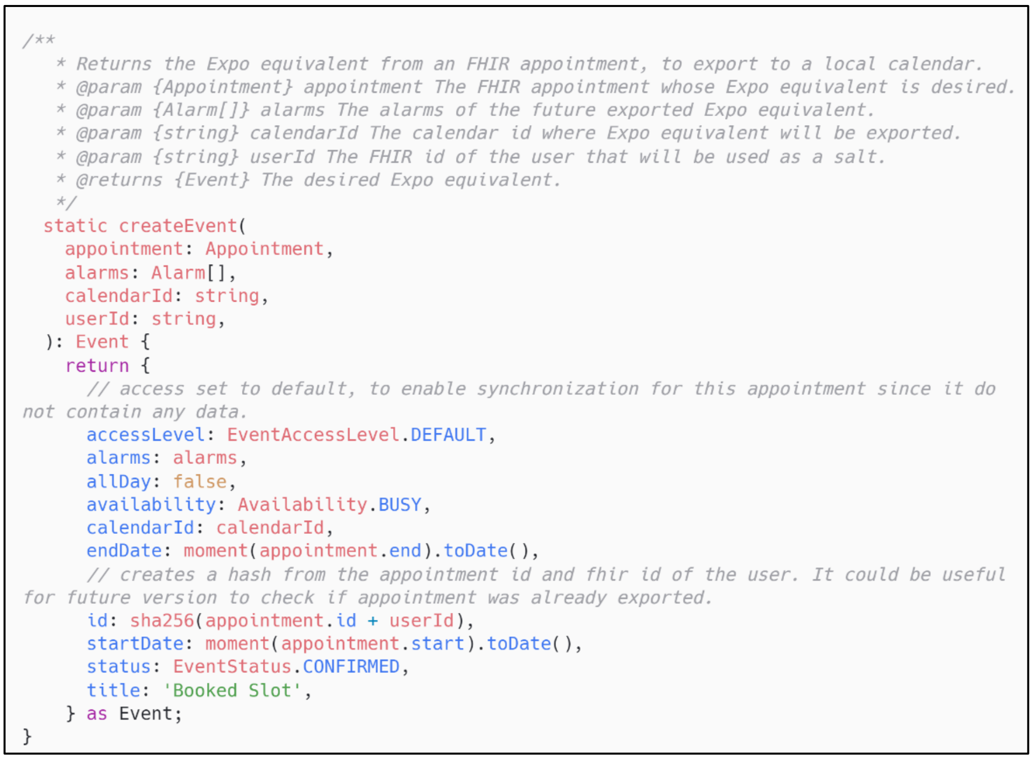
One of the main functionalities of Firecard is the possibility to export an Appointment to a regular calendar without sensitive information. To do so, we must define which data are sensitive, see Figure 1. To read calendars available on the device and write in it, Firecard uses a package called “expo-calendar”. When activating the synchronization in the settings, Firecard will ask the permission to access calendar data. If the permission is granted, the available calendars are read and filtered.



```
async function getCalendars() {
      const {status} = await Calendar.requestCalendarPermissionsAsync();
      if (status === 'granted') {
        const calendarsList = await Calendar.getCalendarsAsync(
          Calendar.EntityTypes.EVENT,
        );
        const desiredCalendars: Calendar.Calendar[] = [];
        for (let i = 0; i < calendarsList.length; i++) {
          if (
            calendarsList[i].isVisible &&
            calendarsList[i].allowsModifications
          ) {
            desiredCalendars.push(calendarsList[i]);
          }
        }
        setCalendars(desiredCalendars);
      }
}
```

Figure 19: Function to ask permission and read available calendars.

Once a calendar is selected by the patient, an appointment can be exported on the display screen. The method createEvent in EventUitls class maps a FHIR Appointment to a calendar Event.



```
/**
   * Returns the Expo equivalent from an FHIR appointment, to export to a local calendar.
   * @param {Appointment} appointment The FHIR appointment whose Expo equivalent is desired.
   * @param {Alarm[]} alarms The alarms of the future exported Expo equivalent.
   * @param {string} calendarId The calendar id where Expo equivalent will be exported.
   * @param {string} userId The FHIR id of the user that will be used as a salt.
   * @returns {Event} The desired Expo equivalent.
   */
  static createEvent(
    appointment: Appointment,
    alarms: Alarm[],
    calendarId: string,
    userId: string,
  ): Event {
    return {
      // access set to default, to enable synchronization for this appointment since it do
not contain any data.
      accessLevel: EventAccessLevel.DEFAULT,
      alarms: alarms,
      allDay: false,
      availability: Availability.BUSY,
      calendarId: calendarId,
      endDate: moment(appointment.end).toDate(),
      // creates a hash from the appointment id and fhir id of the user. It could be useful
for future version to check if appointment was already exported.
      id: sha256(appointment.id + userId),
      startDate: moment(appointment.start).toDate(),
      status: EventStatus.CONFIRMED,
      title: 'Booked Slot',
    } as Event;
}
```

Figure 20: Function that maps FHIR Appointment to calendar Event.

Here, the access level is set to DEFAULT. It could be a nice addition to add a parameter in settings to set this value. Because it determines if the event will be shared if the calendar is shared. This exported event does not contain any sensitive data. That is why that is not a big problem to do so. It is even good because if the patient shares it in a work calendar, colleagues will know that he is not available. The alarms should also be defined in the settings, but due to time limitations it is not the case here. The ID of the exported appointment is a sha256 hash with the appointment ID and the FHIR ID of the patient. This is useful, as it would allow us in a future version to know if an appointment has already been created. And to modify it if the original appointment has been modified. In terms of privacy this is not a problem, as these IDs have no meaning of their own outside MIDATA, and you need the right permissions to see them even if you have access to MIDATA. Furthermore, it's hashed, so there's almost no chance of it being a problem.

3.5.5 Testing

Testing a mobile application automatically is not a basic thing. There are a few frameworks to do this, but it would have greatly increased the workload. Instead, unit tests have been created for the testable parts, i.e., certain utility classes. So, the global testing was done manually, with several accounts, by having other people try it.

3.6 Usability Test

Usability tests were carried out to get feedback on the use of the application. Users of all age groups tested the application. Indeed, it is important that both young and old can use Firecard. It emerged in the first interview that most people do not really care if their sensitive data is shared on foreign clouds. However, some still prefer to write it all down. But all were aware of this issue. The feedback was very good, and only few changes had to be made because of these tests. The participants all found the idea interesting, and the export function is a big plus. If health actors were involved in the development of a project like this, many would use such an application.

4 Discussion

The Firecard application, and the thinking behind it, shows that it is still possible in 2022 to create privacy-friendly applications, while still having the practicality that modern applications offer. That means, cloud backups, communication with other applications or services, while respecting the privacy of the users who are patients. It would have been simpler to create an application that simply allows you to record your appointments in a secure application that does not communicate with the outside world. But this would have raised several problems: How to guarantee the accuracy of the data entered? How to manage the multi-device support? How to have a bidirectional flow of information between doctors and their patients? And above all, what would have been the contribution of such an application compared to a simple calendar or non-clouded diary application? Today the challenge of medical informatics is digitalization, the health sector must evolve to face its difficulties such as costs, errors and an ageing communication model. Firecard aims to offer a solution to these challenges. The development of a privacy-friendly application starts with defining what data will be stored, how it will be processed and by whom and where it will be stored. It is possible to manage patient data in an application in compliance with the Federal Act on Data Protection (as well as the Federal Act on Research involving Human Beings in some cases, in this case it was not). And, by ensuring not to sell the data. But this effort is useless if the cloud platforms and communication technologies used do not themselves comply with these principles. Therefore, a service like MIDATA offers is essential for the development of connected mHealth applications today. The MIDATA cooperative assures in its terms and conditions and privacy statement that it respects the law, and will not sell this information, in addition to guaranteeing a secure service, up to date with the latest standards. This is important because GAFAM provide FHIR cloud infrastructures, but in this case, the data will not necessarily be stored in Switzerland, and therefore fall under the jurisdiction of other laws, such as the Patriot Act in the US. Another significant point that was raised during this project is that patients should be able to control their data and be able to decide for themselves what to do with them. In the current version of Firecard, only the date and time are exported to a conventional calendar, and only if patients decide to do so. But the base idea was to have a solution where patients could choose themselves which data they want to export. Perhaps some patients would like to have the address of their appointment exported as well, to facilitate interaction with a GPS, a voice assistant or a public transport application. It is up to patients to be able to define their own cursor. After talking to several people, most are aware of the problem of saving private data in their Google, Microsoft or other calendars, but they do it because it is convenient for them. Some people said that they write down the address, but not the name of their doctor. Applications like Firecard could solve these problems, especially in a future version, where patients could define this famous cursor.

Using MIDATA implies the use of FHIR for data exchange and storage. But the question is: would there have been any other solution? The answer is yes, if today in the medical sector FHIR tends to become the norm, it has not always been so. In the past, there was HL7 V2 and HL7 CDA, which are also an exchange format, but their use is declining in favour of FHIR. The advantage of FHIR is that from the beginning it was designed to exchange data, but also to store it, and FHIR can perform operations on data at a granular level. Older standards do not support this, that make FHIR unique [4]. The standard provides the basis for a REST API to make direct requests to the server via the HTTP protocol. This is excellent for mobile applications because it requires few resources, and this protocol is well-supported, especially in React-Native. So, on the medical standards side, we can say that FHIR is the best solution, especially since eHealth Suisse is now promoting the use of FHIR, so using this standard is future-proof. However, outside the medical field, there are different standards for appointment management. One is the iCalendar (more commonly iCal) exchange format, which defines the semantics of what an appointment is. But to use it in Firecard for example, one would still have to use an exchange protocol to transmit iCal appointments like SMTP. So, one can see that here FHIR was the simplest and most obvious choice because it defines both the exchange format and the exchange protocol, and MIDATA makes it effortless to use both. However, FHIR is a young standard and there was no guarantee that it would be possible to do such a project with this technology. Firecard shows that it is possible, but there are several shortcomings. Unlike a format like iCal, it is not possible without an extension to know the creator of the event, fortunately MIDATA has included this extension for us. There is no field for a description (not to be confused with the description field, which serves as the title of an event). Firecard uses the comment field for this. But this could cause confusion in terms of interoperability when exchanging with other FHIR ecosystems that would use it not as a description of the event, but as a real comment field. Nevertheless, it is relatively easy to switch from one format to the other. Where FHIR has a significant advantage is that it can contain references to other FHIR elements, such as a laboratory result, as a reason for the appointment. It goes without saying that FHIR is suitable for this kind of project, and even without MIDATA, FHIR would have been used. The Appointment resource is not yet definitive and needs to be improved, feedback would be given on this subject.

As described above, Firecard demonstrates that it is possible to create a system for sharing and viewing medical appointments while respecting patient privacy. The application also demonstrated that FHIR was a suitable choice for this project. Now, would it be possible to see such an application in the next months / years in Switzerland? The answer is unfortunately no. It would be necessary that the primary systems, i.e., the IT systems of the medical practices or hospitals have a connection to MIDATA, which is currently not the case. Moreover, many institutions would have to set up this connector if it existed, and we can already see that the EPR (electronic health records system proposed by Switzerland) is difficult to set up and is not used massively. Speaking of the EPR. Firecard could have been developed using this instead of MIDATA, but there would have been several problems. Even though eHealth Suisse promotes the use of FHIR, the EPR is not a FHIR server. FHIR resources can be stored in the EPR, but it does not have the connectivity that a FHIR server offers. In the latter, the Appointment resource is shared with all participants, so if one of them updates its participation status, all participants will have the updated resource directly. With the EPR, it would have been necessary to invalidate each copy for each participant and then send a new version to all of them. There is a Mobile Access Gateway project that allows you to “simulate” a FHIR server on the EPR, but this involves using transactions from the IHE profile, so you cannot use the FHIR REST API directly. It also does not add the connectivity of FHIR. The EPR can be seen as a static archive, on which new documents are simply stored. In its current state, an application like Firecard could be pushed by associations, for example the Swiss Heart Foundation. But to be used more globally, the EPR would have to evolve, and all health actors would have to participate. At present, nothing is being done in this regard, and the issue of appointments and their confidentiality is not being addressed, according to the eHealth Suisse website. However, we can be sure that hospitals and practices will move towards more connected solutions such as Firecard in the coming years. But given that the EPR does not offer the solutions to achieve this, it is likely that they will use in-house solutions, not very confidential, such as emails for example. Or solutions provided by GAFAMs such as Apple and its CareKit. Therefore, this Firecard project is important because even if in its current state, it does not allow for mass use, it allows us to ask the right questions, at an early stage of the problem. It allows the issues that the health system has in terms of digitalization to be brought to light and lays the foundations for a confidential and effective system.

5 Perspectives

As seen in the results, some elements could not be added in this version, for a next one, it would be interesting to work on it. The idea of using FHIR Schedule would also be a very useful option, as it would allow patients to request an appointment with their doctor directly via Firecard.

At the end of the discussion, we can see that for the moment such an application is not ready to see the light of day. However, with a little more work on it, a good concept could emerge that would lay a solid foundation for the time when healthcare will start to take an interest at this theme.

6 List of illustrations

7 Contents of the table

8 Bibliography

[1] S. McLean *et al.*, ‘Appointment reminder systems are effective but not optimal: results of a systematic review and evidence synthesis employing realist principles.’, *Patient Prefer. Adherence*, vol. 10, no. 1, pp. 479–499, Apr. 2016, doi: 10.2147/ppa.s93046.
[2] J. Chaiwongsai, P. Preecha, and S. Intem, ‘Automated patient appointment reminder for cross-platform mobile application’, pp. 1–6, Oct. 2016, doi: 10.1109/ispacs.2016.7824765.
[3] M. F. Walji and J. Zhang, ‘Human-Centered Design of Persuasive Appointment Reminders’, pp. 236–236, Jan. 2008, doi: 10.1109/hicss.2008.194.
[4] Muhammad Ayaz *et al.*, ‘Correction: The Fast Health Interoperability Resources (FHIR) Standard: Systematic Literature Review of Implementations, Applications, Challenges and Opportunities’, *JMIR Med. Inform.*, vol. 9, no. 7, Jun. 2020, doi: 10.2196/21929.

9 Appendix

You will find the appendices in order at the end of this document.

1. Project Charter
2. Specifications
3. Planning
4. Technical Documentation
5. Usability Test Protocol

10 Declaration of Authorship

I hereby certify that I composed this work completely unaided, and without the use of any other sources or resources other than those specified in the bibliography. All text sections not of my authorship are cited as quotations, and accompanied by an exact reference to their origin.

Place, date: Villeret, 16.06.2022

Signature: [illegible]