Healthcare Transformation with Informatics and Artificial Intelligence
J. Mantas et al. (Eds.)

doi:10.3233/SHTI230436

FHIR DataProvider for ReactAdmin: Leveraging User Interface Creation for Medical Web Applications

Raphael SCHEIBLE[a,1], David ALKIER[b], Justus WENDROTH[b], Julian MAYER[b] and Martin BOEKER[a]
[a]*Institute for AI and Informatics in Medicine, University Hospital rechts der Isar, Technical University Munich, Munich, Germany*
[b]*School of Computation, Information and Technology, Technical University Munich, Munich, Germany*

Abstract. In medical data science, FHIR provides an increasingly used information model, which will lead to the creation of FHIR warehouses in the future. To efficiently work with a FHIR-based representation, users need a visual representation. The modern UI framework ReactAdmin (RA) enhances usability by leveraging current web standards such as React and Material Design. Rapid development and implementation of usable modern UIs is made possible by its high modularity and many widgets available in the framework. For data connection to different data sources RA needs a DataProvider (DP), which maps the communication from the server to the provided components. In this work, we present a DataProvider for FHIR that enables future UI developments for FHIR servers using RA. A demo application demonstrates the DP's capabilities. The code is published under MIT license.

Keywords. FHIR, User-Computer Interface, Medical Informatics Applications

1. Introduction

Data warehouses are one of the foundations of medical data science. To realize such warehouses, i2b2 [1] and transmart [2] are popular applications. In the last years, Fast Healthcare Interoperability Resource (FHIR) gained popularity as a common data format, especially in the context of information models for mobile and web applications as well as medical devices [3]. Besides the reference FHIR server implementation HAPI FHIR [4], other implementations such as Blaze [5] and LinuxForHealth (former IBM) FHIR server [6] were released promising higher performance. In order to provide easy access of data to users like MDs, user interfaces (UI) are required [7]. UIs are constantly evolving and often show much potential in terms of usability in the medical domain [8]. With the medical mesh browser [9], we recently demonstrated that, the usability can be

[1] Corresponding author, AIIM, Klinikum rechts der Isar der Technischen Universität München, Ismaninger Str. 22, 81675 München, Germany; E-mail: raphael.scheible@tum.de.

improved [10] with certain decisions made on used UI technology. The framework ReactAdmin (RA), which uses React and Material Design, was chosen for the UI. It uses DataProviders (DP) to bridge and abstract the communication of its widgets to certain backend technologies. By developing new data providers, new backend technologies can be made compatible with the framework to rapidly create a usable and modern UI. We present a FHIR data provider and show its current capabilities within a demo application.

2. Methods

2.1. DataProvider

Each DP implements specific functions which are internally called by the widgets included in RA to access FHIR resources (cf. Table 1). Additionally, we even implemented a special feature, which allows the FHIR search operators to be used within the filter functionality of RA. Another exceptional requirement of FHIR compared to regular APIs is its characteristic to send sparse responses as answers to a request: for empty fields in a FHIR resource, the server neglects this field in the returned JSON representation leading to a missing key. RA expects the returned fields to be known and fully presented, otherwise the application crashes using its default widgets. One solution is to create customized widgets derived from the default widgets to handle these cases of missing keys. Alternatively, the FHIR Extender can be used, which solves the missing key problem by extending defined FHIR resources to some degree based on a manually set configuration. It populates the expected FHIR expression with empty default values for each required resource. When FHIR resources are sent to the server, they are reduced based on the same configuration. To assure correctness and minimize errors, unit tests were developed for all components of the DP. Additionally, a continuous integration (CI) process was designed which checks these tests to pass in the git repository and can be used to publish versioned releases. The main code of the DP is written in Typescript.

Table 1. List of all required DP functions, its usage meaning and mapping to the FHIR server

Method	Usage	Mapping

getList	Search for resources	GET {baseURL}/{res}?_summary=count&{param=value}*
		 GET {baseURL}/{res}?{param=value}*

getOne	Read a single resource by id	GET {baseURL}/{res}/{id}
getMany	Read a list of resource, by ids	GET {baseURL}/{res}/_id={val1,val2,val3}
getManyReference	Read a list of resources related to another one	GET {baseURL}/{res}?target=value&{param=value}*
create	Create a single resource	POST {baseURL}/{res} (id is returned by server in header)
update	Update a single resource	PUT {baseURL}/{res}/{id}
updateMany	Update multiple resources	PUT {baseURL}/{res}/{id} (multiple times)
delete	Delete a single resource	DELETE {baseURL}/{res}/{id}
deleteMany	Delete multiple resources	DELETE {baseURL}/{res}/{id} (multiple times)

2.2. Demo Server

For demo purposes, we set up a FHIR server environment. We picked the well tested and established LinuxForHealth FHIR server. For easy deployment docker was used. We generated a plausible FHIR test dataset using Synthea [11] for the test application. The test dataset consists of Patient resources representing 250 alive and 4 deceased patients in the age of 0-100 years. Data were generated with Location set to the state of Messachusetts, all disease modules contained in Synthea used and a patient history of up to 10 years allowed, leading to 10113 Providers, 82760 Observations, 11212 Conditions and 463 Organizations and Practitioners. The server was filled with the data via a python script.

2.3. Demo Application

The demo application is a full featured web application with authentication. Therefore, RA offers the interface for an AuthProvider, for which we implemented one specifically for the LinuxForHealth FHIR server. The rest of the application was mainly realized by the standard RA components List, Edit and Show to visualize the resources from the FHIR server. Inside the List component we used different RA components such as ArrayField to show the telefone of a Patient, DateField to show the birth date of a Patient or the TextField to display the status of an Observation. Within a List component, pagination allows the dataProvider to only return a subset of resources to be rendered. Users can change the page to browse through all resources.

3. Results

3.1. DataProvider

The DP supports the FHIR standard and can parse arbitrary FHIR resources. Further, it can make use of the FHIR search feature to apply filters to specific fields. The FHIR Extender works with simplex structures, but has problems with nested and more complex FHIR structures using 1:N relations. The unit tests cover 100% of functions and 99.20% of all code. Currently, the DP fully supports the LinuxForHealth FHIR server and the HAPI FHIR server for read operations. It is released under the MIT license at https://gitlab.com/mri-tum/aiim/libs/ra-data-fhir.

3.2. Demo Application

The demo application starts with a Login page which is automatically displayed due to the plugged in AuthProvider. Further, 11 views for show, edit, create, and list for multiple FHIR resources are implemented (cf. Table 2). Two of the list views are enabled to filter for certain criterias. To handle missing keys, the FHIRExtender is applied by default and in one case we used one custom component.

Table 2. List of implemented views of the demo and its properties.

Resource	Show	Edit	List	List filter	Create
Patient	+	+	+	Patient name	+
Observation	+	-	+	Patient ID and date	-
Organization	+	+	+	-	-
Practitioner	+	-	+	-	-

4. Discussion

Here we present a DP as a first step towards using RA in combination with FHIR servers in a demo application. However, in a production scenario, we would recommend more sophisticated authorization mechanisms which could be realized using KeyCloak. Further, currently the demo misses delete operations, as for many FHIR structures it would require recursive deletion. FHIR resources are validated only on server side, since validation inside the DP would significantly increase its complexity. Thus, no explicit FHIR profiles are implemented on client side, but implicitly by the usage of the resource properties of the application. We solved the issue of missing keys by the FHIR Extender. However, its default values should be selected with caution, as otherwise it could lead to missing data due to the reduction step while sending data to the server. In its current state, the FHIR Extender is limited to 1:1 relations for nested structures. Here, more work is required to make it universally compatible. The present DP is limited to run flawlessly in combination with the LinuxForHealth FHIR server. Compatibility with the HAPI FHIR server is only given for read operations. In the future, support for more FHIR servers is desired.

5. Conclusion

With this work, we have taken the first step to create UIs with RA on top of FHIR servers. The problem of sparse FHIR resources was solved by extending and reducing it using FHIR Extender. Our demo application demonstrates the capabilities and potential of this technology.

Acknowledgements

Supported by the German Ministry for Education and Research (FKZ 01ZZ1804A).

References

[1] Murphy SN, Weber G, Mendis M, Gainer V, Chueh HC, Churchill S, et al. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). J Am Med Inform Assoc JAMIA. 2010 Apr;17(2):124–30.

[2] Scheufele E, Aronzon D, Coopersmith R, McDuffie MT, Kapoor M, Uhrich CA, et al. tranSMART: An Open Source Knowledge Management and High Content Data Analytics Platform. AMIA Jt Summits Transl Sci Proc AMIA Jt Summits Transl Sci. 2014;2014:96–101.

[3] Lehne M, Luijten S, Vom Felde Genannt Imbusch P, Thun S. The Use of FHIR in Digital Health - A Review of the Scientific Literature. Stud Health Technol Inform. 2019 Sep 3;267:52–8.

[4] HAPI FHIR - The Open Source FHIR API for Java [Internet]. [cited 2023 Mar 13]. Available from: https://hapifhir.io/

[5] Blaze [Internet]. Samply; 2023 [cited 2023 Mar 14]. Available from: https://github.com/samply/blaze

[6] LinuxForHealth/FHIR: The LinuxForHealth FHIR® Server and related projects [Internet]. [cited 2023 Mar 14]. Available from: https://github.com/LinuxForHealth/FHIR

[7] Prasser F, Kohlbacher O, Mansmann U, Bauer B, Kuhn KA. Data Integration for Future Medicine (DIFUTURE). Methods Inf Med. 2018 Jul;57(S 01):e57–65.

[8] Bin Azmat S, Hanif MK, Zia U, Rehman AU, ul Haq I, Shahzad I. Cognition based user interface design for healthcare systems. Int J Comput Sci Netw Secur. 2019;19:177–85.

[9] Scheible R, Strecker P, Yazijy S, Thomczyk F, Talpa R, Puhl A, et al. A Multilingual Browser Platform for Medical Subject Headings. In: Informatics and Technology in Clinical Care and Public Health. IOS Press; 2022. p. 384–7.

[10] Strecker P, Boeker M, Buechner S, Scheible R. Usability Evaluation of a Modern Multilingual MeSH Browser. In: Advances in Informatics, Management and Technology in Healthcare. IOS Press; 2022. p. 37–40.

[11] Walonoski J, Kramer M, Nichols J, Quina A, Moesel C, Hall D, et al. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. J Am Med Inform Assoc JAMIA. 2018 Mar;25(3):230–8.