A MOBILE APPLICATION ACCESSING PATIENTS' HEALTH RECORDS THROUGH A REST API

How REST-Style Architecture can Help Speed up the Development of Mobile Health Care Applications

Francois Andry, Lin Wan and Daren Nicholson
Axolotl Corp., 160 West Santa Clara Street, San Jose, CA 95113, U.S.A.

Keywords: Mobile Application, Health Records, RHIO, Lab Results, HL7, REST API, Web Services, JAX-RS, Security.

Abstract: Mobile devices offer new ways for users to access health care data and services in a secure and user-friendly environment. These new applications must be easy to create, deploy, test and maintain, and they must rely on a scalable and easily integrated infrastructure. In this paper we present the motivations and technical choices for creating a REST API integrated with a mobile application (iPhone/iPad) that offer physicians, access to their patients' health records via a community, regional or state Health Information Exchange (HIE). We describe the architecture of the system, including how we address security and privacy concerns, the REST API operations and HL7 subset data format used for lab results and observations. We also explain why the early use of unit tests and integration tests were essential to the success of the project.

1 INTRODUCTION

In the ambulatory health care environment, providers spend the majority of their time in an examination room with patients. Although some clinics have installed personal computers in the exam room for use at the point of care, many physician practices have yet to do so or have no such intention. Reasons for not installing PCs in the exam room include (among others) lack of space, security concerns, and cost. Often, clinics have PCs installed outside of the exam room to be used for encounter documentation or health history research (i.e., reviewing the patient's health records). This physical setup is often satisfactory for providers to complete their documentation needs. Providers often scratch rough notes on paper during an encounter, then dictate or type their notes after the visit has ended. The absence of computers in the exam room, however, is a disadvantage for research activities. Frequently, after listening to the patient's verbal health history, a provider wishes to read past records. If those records are in an electronic format, it is optimal to access those records at the point of care (i.e., in the exam room) (Shiffman et al., 1999). Thus, computer devices that are smaller and more mobile than a PC (e.g., smart phones, PDAs, tablets) would be the optimal hardware choice to access these electronic records (Sammon et al., 2006; Kumar et al., 2009). Given that many physicians carry smart phones, such mobile devices would be the ultimate tools to look up patient records (Watson, 2006).

Over the years, our group has developed advanced Clinical Networking™ solutions for hospitals, RHIOs and state-wide health information exchanges (HIE). We have created a backend infrastructure accessible for integration as software-as-a-service (SaaS), most of them relying on Simple Object Access Protocol (SOAP) APIs. One of the issues we have encountered when creating an architecture for mobile application was the type of API to use between backend servers and the new mobile clients.

In this paper, we explain the challenges (Pabllo et al., 2008) faced when creating a new high performance API accessible by mobile health care applications for physicians and medical personnel. We describe the technical choices that we have made to simplify and to speed up the development, management, extension and maintainability of the features involved in such applications and the associated service layer.

Andry F., Wan L. and Nicholson D..
A MOBILE APPLICATION ACCESSING PATIENTS' HEALTH RECORDS THROUGH A REST API - How REST-Style Architecture can Help Speed up the Development of Mobile Health Care Applications.
DOI: 10.5220/0003129600270032
In *Proceedings of the International Conference on Health Informatics* (HEALTHINF-2011), pages 27-32
ISBN: 978-989-8425-34-8

2 MOBILE APPLICATION

The initial mobile application that we built on top of Elysium Virtual Health Record (VHR) infrastructure, only displays lab results. Other types of data (vital signs, radiology and transcribed reports, medications, etc.), which are currently accessible via a web-based application, will be offered at a later time. The usage scenario for the mobile application is as follows:

1. Physician logs in using her/his credentials.
2. After successfully logging in, the physician is presented with a list of patients that he/she has explicit consent to view. In addition, there is also a search box that will allow the physician to manually search for patients.
3. The physician chooses the patient of interest on the patient list (or search results).
4. Once the patient is selected, the physician sees a list of available lab results for that patient.
5. The physician selects a lab result of interest by tapping it. This opens the lab result and displays all of its details in a detail screen, with the option of going back to the previous screen to retrieve another lab result.

2.1 Security Concerns

In order to minimize security risks and to comply with HIPAA security regulations, we chose not to store any patient, login or password data on the mobile device preventing non-authorized users from gaining any access to a patient's personal health information (PHI).
Access to the patients' health records are provided after the user (typically a doctor) enters his or her login and password on the client application. These are sent to the server via an application programming interface (API). After authentication is successful, the server sends back a security token that the client must reuse for each subsequent request to the server API. Initially all users (physicians) have the same access control to the data as long as the patients have given consent to specific physician practices. As a result, the authorization function is very simple.

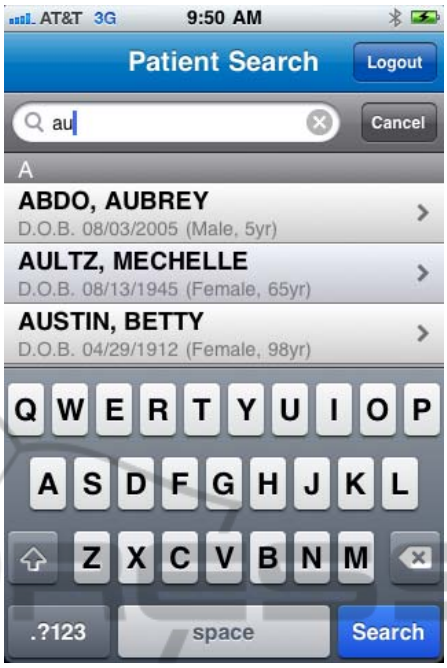

Figure 1: Patient search (iPhone application).

2.2 Client Platforms

The initial platforms chosen to deploy our mobile applications are the iPhone and iPad. Additional platforms (Google Android OS, Motion BlackBerry, Windows Mobile OS) are planned for future releases.
Building, deploying and maintaining stand-alone applications for mobile platforms is different from building browser-based web applications. Each platform has its own design, development and deployment process and tools.
Since the development of client applications on different mobile platforms requires more time than creating web applications for a handful of browsers, it is important to minimize the complexity of the integration with the back-end services and to try to decouple the development and maintenance of the client- and server-side components.

3 BACKEND INFRASTRUCTURE

For over 15 years, Axolotl (now part of Ingenix) has developed and offered web-based applications to access patient records. Until recently, these web-application components were either tightly coupled with the backend data sources or were accessing data through Simple Object Access Protocol (SOAP) based web services. In health care, SOAP is often

the top choice when it comes to web services. The advantages of SOAP are:

- Type checking (via WSDL files)
- Availability of development tools
- Suitability for securely transferring large numbers of data

Even though it was tempting to reuse the existing SOAP APIs to leverage the existing infrastructure and avoid new development on the server side, we decided to create a more appropriate architecture that would be easier to integrate and extend as more types of data are added to the mobile health care applications.

4 VHR REST API

The Representational State Transfer (REST) architecture (Fielding 2000) is an alternative to SOAP and offers certain characteristics that were relevant to our use case:

- Lightweight and easy to build
- Extensible
- Scalable
- Easy to debug (human readable results)

SOAP based Web Services offer no separation of concern, since the supported programming model does not separate network centric operations from local operations (Landre, Wesenberg, 2007). On the other hand with REST APIs, the separation of concern is clear. All resources are accessible using the same protocol (HTTP). Also with SOAP, interoperability problems can occur when native types are present in the interface, whereas REST is simpler as it completely constrains the set of operations (Pautasso et al., 2008).

4.1 API Design and Development

With REST we are able to dynamically build unique URLs to represent remote health records objects as needed. The mobile application sends HTTP requests over Secure Sockets Layer (SSL) to obtain a JavaScript Object Notation (JSON) of patient demographic info (DOB, name, etc) or health records. JSON was chosen because, as a compact data format, it offers better performance compared to the complexity of an XML representation. In addition to this, requests coming back from the REST API are compressed using GZIP, which further improves the performance between the server and the client.

The API has been built using JBOSS JAX-RS RestEasy library, which leverages Java annotations,

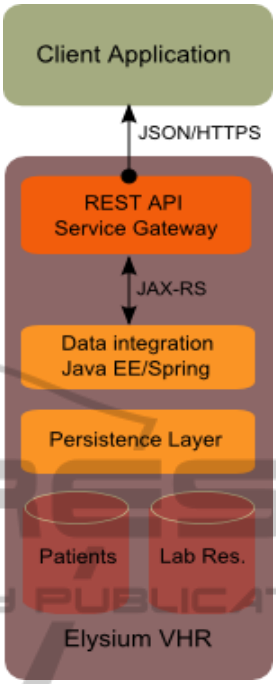

Figure 2: Integrated Mobile/REST API architecture.

making the definition of the resource mapping easy to do in a very declarative manner. We use the Jackson JSON/Java library to marshal and unmarshal java objects into JSON format. The integration of the API layer with the backend data source, as well as other services such as configuration, is done using the Java Spring framework.

To define the API dependencies, modules, libraries and to build the web archive artefact (war), we use Apache Maven 2. Throughout the development cycle, we have used a continuous integration process with unit tests (JUnit).

The overall project has been managed through an agile/SCRUM process, with iterative, incremental development sprints, each lasting three weeks.

4.2 API Operations

With REST, the identification of the resources (user, patient, lab results, etc.) is straightforward. There is no need for the client to create complex request envelopes to query the server.

The initial features of the mobile client is to access the health record in a read-only mode; no data is created or modified. As a result, most of the operations use GET HTTP methods, with some

restrictions for certain operations, where parts of the PHI may be sent by the user back to the server. To be compliant with HIPAA security regulations, a POST method is used and the sensitive information (e.g., a query on a patient name) is passed in the body of the request as a JSON object over an SSL connection.

Table 1: Health Record API operations examples.

POST /users/tokens	authentication for a user
GET /users/{user-id}/patients	retrieve patients
GET or POST /users/{user-id}/patients/{patient-id}	profile of a specific patient
GET /users/{user-id}/patients/{patient-id}/lab-results	lab-results of a patient

All request and response headers have a content type "application/json," which means that complex queries and responses are in the form of JSON arrays or JSON objects. In addition to this, the requests coming from the client need to provide the security token provided initially by the server as a cookie in the request header.

Because the REST API is stateless, queries for large amounts of data which require more than one request - have to provide a limit (maximum) for the number of resource items and an offset where to start the query to avoid returning the same set of data. These are defined as string parameters or posted queries.

Figure 3: VHR patient's lab results summary (iPhone).

Example of a JSON query object for patients:

```
{"query":
   {"family":"...",
    "max":N,
    "offset":M}
}
```

Example of a JSON response object for patients:

```
{"patients":
    {"list":[list of patients],
     "count":X,
     "offset":Y,
     "remain":Z}
}
```

Heath Record data can be very complex. For example in our data source, data is often stored as HL7v2 fields. The REST API uses an HL7v2 to HL7v3 transformation library to access Java canonical representation of the observations and unmarshal these plain Java objects (POJO) into an HL7v3 JSON representation of the observation.

Example of a JSON response object for lab results:

```
{"lab-results":
      {"list":
            [{"lab-result":
                  {"entry":"...",
                   "facility":"...",
                   "normalcy":"...",
                   "orderedBy":"...",
                   "status":"...",
             "subject":"...",
                   "urgency":"..."}},
              {"lab-result":
                  {"date":"...",
                  ...}}, ...],
      "count":{count},
      "offset":{offset},
      "remain":{remain}}
```

Figure 4: VHR patient's lab results details (iPhone).

HL7 V3 Lab result entry (JSON format):

```
{"entry":{
   "organizer":{
        "code":{"displayName":"..."}},
        "components":[
            {"component":{...}},
                {"component":{...}},...],
        "notes":[...]}}}
```

HL7 V3 Lab result component (JSON format):

```
{"component":{
  "observation":{
     "code":{"displayName":"..."},
     "effectiveTime":{"value":"<ISO-8601>"},
     "value":...,
     "interpretationCode":{"code":"..."},
     "referenceRange":{
         "observationRange":{...}},
     "notes":[...]}}}
```

5 INTEGRATION AND TESTS

The REST API is very easy to test using HTTP requests from simple command tools such as cURL or a web browser. Early in the project we were able to create and deploy, in a few days, a full working prototype of the API that was producing mocked-up resources so that mobile clients were able to retrieve sample data in the same fashion as the final API production version.

During development we used a very small footprint embedded java web server (TJWS) that was running JUnit integration tests every time we were compiling the project. As a result, the REST API was always operational and ready to be deployed as a war file to the staging server when necessary.

On the client side, the very simple host, port and base URL configuration is necessary to switch between a staging and production server, which makes development, testing and deploying extremely easy.

The first version of our mobile health record will be available late 2010 as a pilot program for a small group of physicians participating in the New York Rochester Regional Health Information Organization (Rochester RHIO). It has already been tested by the technical services of the New York Rochester RHIO and we have incorporated changes in the iPhone interface and REST API format to support the variability of real world lab results (which sometimes can be either incomplete or contain badly formatted HL7 data sets).

Initially only the records of patients who have provided consent will be accessible by the mobile application. This represents, in Summer 2010 290,000 patients from more than 140 practices.

6 CONCLUSIONS

In this paper, we have presented the advantages of using a REST architecture for designing a lightweight and scalable API that is extremely easy to build, integrate, test, extend and maintain. We were able to create a working API prototype in a matter of days and a full functioning set of sophisticated health record web services accessible by our mobile client application in few weeks.

In the next few months, we will be able to gather user feedback from a large group of users (physicians and nurses) and statistics about usage that will be very valuable to improve the performance of our health record REST API and the user experience of the mobile application. Also the audit of server logs for the REST API and the use of data mining tools will be valuable to see performance bottlenecks and usage trends. Furthermore not only are REST APIs are particularly suitable for fast and loosely-coupled solution integration such as mobile applications, but can also be used in health care for portal and mash-up applications as well.

ACKNOWLEDGEMENTS

We are very grateful to Ted Kremer, Gloria Hitchcock and LaRon Rowe from the Rochester RHIO who have initiated this project and for their early feedback on the prototype. Our appreciation to David Stanfill and Nick Fisser and their team from Remedy Systems who helped us develop the iPhone mobile client. We would like to acknowledge the supervision and guidance provided by Anand Shroff. Thank you also to Terena Chinn-Fujii, Igor Kosoy, Greg Kuhnen, Ravi Luthra, Brian Schott, Neal Schultz, Nick Radov, Nicole Spencer, Dennis Stratford, Sean Smith, Tom Wilson, Eileen Xie, Oleg Zakharov and the Axolotl development team for their technical contributions, comments and support throughout this project.

REFERENCES

Fielding R., 2000, Architectural Styles and the Design of Network-based Software Architectures. *Doctoral dissertation*, University of California, Irvine.

Fielding R., Taylor R., 2002,. Principled Design of the Modern Web Architecture, in TOIT 2002: *ACM Transactions on Internet Technology*, pp. 115–150.

HIPAA, The Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy and Security Rules. http://www.hhs.gov/ocr/privacy/.

Kumar et al., 2009, ELMR: Lightweight Mobile Health Records: A Study of Access Control. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*.

Landre E., Wesenberg H., 2007: Rest versus soap: as architectural style for web services. In *5th International OOPSLA Workshop on SOA & Web services Best Practices*, 2007.

Luo J., 2008. Mobile computing in healthcare: the dreams and wishes of cliniciansm, In *HealthNet '08: Proceedings of the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*.

Pabllo C., Soto R., Campos J., 2008. Mobile Medication Administration System: Application and Architecture. In *EATIS '08: Proceedings of the 2008 Euro American Conference on Telematics and Information Systems*.

Pautasso C., Zimmermann O., Leymann F., 2008. RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. In WWW2008: *17th International World Wide Web Conference, Beijing, China*.

Sammon et al., 2006, MACCS: Enabling Communications for Mobile Workers within Healthcare Environments. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*

Shiffman et al., 1999, Pen-Based, Mobile Decision Support in Healthcare. In *SIGBIO Newsletter, Volume 19 Issue 2*.

Watson M., 2006, Mobile Healthcare Applications: A Study of Access Control. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust*.