rdrf Documentation

Release 1.1.1

CCG

March 09, 2016

Contents

The Rare Disease Registry Framework (RDRF) is an open source tool for the creation of web-based Patient Registries, which delivers the features that you and/or your organisation currently require. The RDRF is flexible, and your Registry can be customised and enhanced to evolve as your needs change. The RDRF empowers Patient Organisations, Clinicians and Researchers to create and manage their own Patient Registries, without the need for software development.

The RDRF is unique in that data entry forms and questionnaires are based on reusable data element definitions (called *Data Elements (DEs)*), which can be created and/or loaded into the system at runtime. This means that registries can be created and modified without changes to the source code. The RDRF has been developed at the Centre for Comparative Genomics, Murdoch University, Western Australia in partnership with the Office of Population Health Genomics, Department of Health Western Australia.

CHAPTER 1

Do you need a patient registry for your department, clinic or community?

The RDRF enables the rapid creation of Registries without the need for software development through the following key features:

- Dynamic creation of *Registries, Forms and Sections* (comprised of forms, sections, and *Data Elements (DEs)*) at *runtime*
- Reusable Components (*Data Elements (DEs)*) allows DEs to be used in multiple registries
- Patients can be defined once, and belong to several registries
- Multiple levels of access are supported (e.g. patient, clinician, genetic, and curator roles)

The RDRF can be used to create different types of registries, such as a Contact Registry or a more complex registry with the ability to restrict Forms to certain groups of users. Please see this video for a quick demonstration.

CHAPTER 2

Are you ready to create your own Patient Registry?

A Demo Site is available for you to try out online. Different levels of access are available, including admin, data curator, genetic staff and clinical staff:

- admin username and password: admin
- data curator username and password: curator
- genetic staff username and password: genetic
- clinical staff username and password: clinical

Screencasts are available to talk you through the creation of a Registry:

- Training videos to create a Registry as an admin user

If you prefer to read:

- Here's some text explaining how to create a *registry in 10 minutes*

CHAPTER 3

Third-party Libraries

- HGVS - Python package to parse, format, and manipulate biological sequence variants according to recommendations of the Human Genome Variation Society. Project URL

CHAPTER 4

Links

Demo Site

Source Code

RDRF Google Group

Documentation

CHAPTER 5

Citations

Bellgard, M.I., et al., Second generation registry framework, Source Code for Biology and Medicine, 2014. 9:14.

Bellgard, M., et al., Dispelling myths about rare disease registry system development, Source Code for Biology and Medicine, 2013. 8: 21.

Bellgard, M.I., et al., A modular approach to disease registry design: successful adoption of an internet-based rare disease registry. Hum Mutat, 2012. 33(10): p. E2356-66.

Rodrigues, M., et al., The New Zealand Neuromuscular Disease Registry. J Clin Neurosci, 2012. 19(12): p. 1749-50.

CHAPTER 6

Documentation Content List:

6.1 Creation of a Registry: Workflows for Design and User Modes

6.1.1 Overview

There are two basic modes in which RDRF is intended to be used: Design mode (for creating new registries) and User Mode (for adding/editing and viewing patient data in a registry that has been created).

Only administrators can add new registries, forms, sections or data elements.

Important Point: Once data elements and sections are created they can be used by more than one registry. But altering a data element or section that is in use will immediately affect any registries which use it.

Additionally, a public interface is provided when a registry form is nominated to be a questionnaire. See *questionnaires*.

6.1.2 Roles

Each workflow in RDRF is intended to be performed by a user with a distinct role.

6.1.3 Admin

- Can do anything
- Create and modify registry definitions
- Import and Export registries
- Manage user permissions

6.1.4 Curators

Manage membership of patients in working groups within a registry

6.1.5 Clinicians

Enter and view data on patients in a working group within a registry

6.1.6 Design Mode Workflow

6.1.7 Modelling

1. Do this first on pen and paper!
2. Gather requirements of the data fields (*"DEs"*) required
3. For each data field required, decide its datatype. If a field is logically a *range*, work out the allowed *permitted values*. Depending on the datatype, decide any validation rules for a numeric or float data type (max and/or min), and for a string data type, the maximum length or pattern. Decide if any Derived Data Elements (calculated data type) are required.
4. Split them into logical groups (*sections*). Decide whether a section might be multiple
5. Portion related sections into *forms*
6. If a questionnaire is required for the registry, nominate a single form as a questionnaire

6.1.8 Creating a Registry

Assuming all *Data Elements (DEs)* have already been created

1. Admin logs in and navigates to "Registries" from "Settings"
2. Admin clicks on green "Add" button
3. Admin fills in Name, code and description of registry (code must be unique and not contain spaces)
4. Admin pastes a html splash screen into the Splash screen field (this will be linked to on the main page)
5. Admin navigates to "Registry Forms" from "Settings" and for each desired form in the registry, clicks the green "Add" button.
6. Admin Selects the registry just created from the drop down list
7. Admin enters a name into the name field (this name will appear on the form, eg "Physical Info")
8. Admin enters a comma-separated list of form section codes (E.g. "FHPhysicalSection,FHPersonalitySection,FHAmbulatorySection" (Note: The codes should be unique and have no spaces - no quotes! - prefixing with registry code is conventional but advised). If the form is intended to be a public questionnaire form, check the questionnaire checkbox
9. Save the form definition
10. For each section referred to in the comma separated list, add a section object by navigating to "Sections" from "Settings"
11. Click the green "Add" button and enter the section code (used in the form definition)
12. Enter a display name for the section (this will appear on the form above the fields defined for the section)
13. Enter the DE codes of any fields required in the elements list (as a comma-separated list) E.g. "CDEName,CDEAge,CDEHeight" (Note- The system will check whether any entered DE codes exist when the section object is saved - if any DE code cannot be found in the system, the section object will not be created)

6.1.9 User Mode Workflows

6.1.10 Adding a user (curator or clinician, or genetic staff)

1. Admin logs in

2. Clicks on “Users” from the “Menu” button
3. Clicks Add User button on right
4. Enters Username and password
5. Clicks Next
6. Enters personal information
7. Checks “Staff Status”
8. Control-Clicks Working Group Curators for curator (or Clinical Staff for clinician, or Genetic Staff for genetic staff)
9. Control-clicks the required working groups and registries (if more than one)
10. Clicks save

6.1.11 Adding a Patient to a Registry

1. Curator or clinician logs in
2. Click “Add Patient” (or click on the Patients name to edit patient)
3. Control-click on each registry that is listed that you would like the patient to be a member of (NB. If a clinician or curator has access to only one registry, it will already be assigned)

6.1.12 Entering (and viewing existing) Demographic Data for a Patient

1. Login as a clinician
2. Click the Patient’s name in the Patient column of the Patient List
3. Edit contact details for the patient
4. Click Save button

6.1.13 Changing Working Group for a Patient

1. Login as curator
2. Click the Patient’s name in the Patient column of the Patient List
3. Select required working group (NB. workings group in the dropdown will only be those for which the curator has access)

6.1.14 Entering / editing existing Clinical Data for a Patient

1. Login in as a clinician
2. If clinician has access to more than one registry a drop down of registries is shown in the search area, otherwise no registry dropdown will appear and all operations will occur in the one registry
3. Click the “Show Modules” button in the patients list for the required patient - a pop up of available forms will appear (except if there is only one defined clinical data form)
4. Click the desired clinical data entry form
5. The screen will show the required form

6. Edit and click Save

6.1.15 Approving/Rejecting a Questionnaire response

1. Curator or clinician logs in.
2. Click "Questionnaire Responses" under "Menu"
3. Click "Review" under "Process Questionnaire" to approve/reject a questionanaire
4. **User reviews information in the submitted form and clicks approve (or reject)**
    - If approve is clicked, a new patient will be created in the registry and working group indicated in the form
    - If reject is clicked, no patient record will be created

6.1.16 Adding a new working group

1. Admin logs in
2. Click on "Working Groups" under the "Menu" button
3. Click the green "Add" button
4. Enter name and save

6.1.17 Changing the Working Groups of a Curator

1. As an admin, click on "Users" under the "Menu" button
2. Click on the username of the curator required
3. Control-click (command-click for Mac) on each working group in the combo box required for that user (a curator in 2 working groups will see patients in both groups)
4. Click the Save button

6.1.18 Assigning a curator (or clinician) to a registry

1. As an admin, login and then click on "Users" under the "Menu" button
2. Click on the username of the user required
3. Control-click (command-click for Mac) on each registry the user is meant to have access to
4. Click the Save button

6.1.19 Adding Genes

1. Admin logs in
2. Click on "Genes" under the "Menu" button
3. Click on "Add" and add details
4. Click Save

6.1.20 Adding Laboratory

1. Admin logs in
2. Click on “Laboratories” under “Menu”
3. Click on “Add” and add details
4. Click Save

6.2 RDRF Graphical User Interface

RDRF is a Django web application and accessed via any web browser (f.g. Chrome or Firefox).

The system uses the standard Django views (generic class-based views) with bootstrap-styling to provide access to the application.

As an admin user, regularly used links are links are accessed via “Settings”, while the “Admin Page” contains the complete list of links.

6.2.1 Admin Page

The Admin Page consists of:

6.2.2 Patient List

Add or edit patient information (contact details)

6.2.3 Doctors

Add or edit Doctors

6.2.4 Reports

Access Reports that have been defined with the Explorer tool

6.2.5 Users

Add or edit users

6.2.6 Genes

Add or edit genes

6.2.7 Laboratories

Add or edit laboratories

6.2.8 Registries

Add or edit registries

6.2.9 Registry Form

Add or edit forms (created forms are accessed under 'Modules' in the Patient List)

6.2.10 Sections

Add or edit sections (Sctions are compiled into Forms)

6.2.11 Data Elements

Add or edit CDEs (CDEs are compiled into sections)

6.2.12 Permissible Value Groups

Add or edit PVGs

6.2.13 Permissible Values

Add or edit PVs (assigned to a pemritted value group)

6.2.14 Groups

Administer available permissions for user groups (working group curators, clinicians, etc)

6.2.15 Importer

Import a registry definition (yaml file)

6.2.16 Demographics Fields

Configure Demographics fields to be hidden or readonly by registry and group

6.2.17 Working Groups

Add or edit working groups (e.g. States, Hospitals, Clinics)

6.2.18 Questionnaire Responses

Questionnaire responses are stored here when submitted by a patient, awaiting validation or rejection by an admin or curator

6.3 Registries, Forms and Sections

6.3.1 Registry

In RDRF, a "registry" is just a named collection of forms. A Registry is created by an admin user by selecting "Registries" from "Settings" and clicking "Add".

A Registry must have a non-blank name and a code (which should not contain spaces).

6.3.2 Forms

A form is a single "screen" of information in RDRF and consists of a group of named Sections. To create a form, select "Registry Form" from "Settings" and clicking "Add".

Form names should not include spaces (e.g. ClinicalData), however the form name will appear as Clinical Data if captial letters are used for separate words.

A form is defined by the following features:

6.3.3 Registry

A link to the Registry that owns the form. A form can only exist in one registry at a time.

6.3.4 Name

The display name to use (this will appear in the "Patient Listing" under "Modules".

6.3.5 Sections

The section *codes* that comprise this form, listed in a comma-separated list.

Example

- SECTIONA,SECTIONB

6.3.6 Is Questionnaire

A checkbox to indicate whether this form is exposed as a public questionnaire.

Questionnaire forms are not loaded from the dashboard but access via an autogenerated URL
/<regcode>/questionnaire

An autogenerated approval form (for curators) is created also.

6.3.7 Sections

A section is a named group of fields that can be inserted into a Form.

Sections consist of:

6.3.8 Code

A section must have a non-blank code (no spaces) which is just a text value. Section codes must be unique. The code of a section is used to refer to it, when used in a Form.

Example: “FHSECTION34” or “SEC001”

6.3.9 Display Name

A string which will be displayed on the form to mark the start of the section.

Example: “Physical Characteristics” , “Contact Information”

6.3.10 Elements

The *DEs* codes of any fields comprising that section.

This *must* be a comma-demlited list of *DE codes*.

Example: “CDEName, CDEAge”

(This would mean that the section would consist of two data input fields - one defined by the *Data Element* with *code* “CDEName” and one defined by the Common Data Element with code “CDEAge”)

NB. Spaces between codes is *not* significant.

6.3.11 Allow multiple

A boolean flag indicating whether multiple entries of this section can added (or removed) on the form.

Example

A friends section might be defined with cdes for name and email address etc. Marking the section as “allow multiple” would cause an “Add” and “Remove” button to be added to the form so multiple friends could be added (or deleted.)

6.3.12 Extra

If allow multiple is checked, this causes extra (blank) copies of the initial section to be displayed.

6.4 Data Elements (DEs)

RDRF allows creation of resusable fields, which can be dropped into the definition of sections of *forms*, simply by entering their code into the elements field of the section definition (in a comma separated list).

A DE is created by an admin user navigating to “Data Elements” in “Settings”

DEs can be shared by all registries created within the framework. A DE definition consists of:

6.4.1 Code

A DE must have a *globally unique code* (e.g. CDEAge, CDEInsulinLevel) which must not contain a space.

A meaningful code prefixed with CDE or the Registry Code is recommended.

6.4.2 Name

A non blank "name" must also be entered, which will be used as the label of the component when it appears on the form.

6.4.3 Desc

Origin of the field if externally loaded.

6.4.4 Datatype

Each cde must have a data type specified by a text descriptor. Currently this descriptor is specified as free text although this may change.

The allowed *Datatypes* are as follows (NB. These are the literal words to type into the datatype field, *except* for ComplexField)

- string
- integer
- alphanumeric
- boolean
- float
- range
- calculated
- file
- date
- ComplexField

6.4.5 Pv group

IF a range, select the desired *permitted value group* here.

6.4.6 Allow multiple

IF a range, checking this box will allow multple selections to be chosen from the range.

Example

- Brands of cars owned
- Medications taken

6.4.7 Max length

IF a string value, the maximum number of characters allowed.

6.4.8 Max value

IF an integer or a float value, the maximum magnitude allowed.

6.4.9 Min value

IF an integer or a float value, the minimum magnitude allowed.

6.4.10 Is required

A check box indicating whether this field is mandatory (any datatype)

6.4.11 Pattern

IF a string value, a regular expression used to indicate admissible values (note these are always case sensitive in the current version).

6.4.12 Widget name

The name of a custom widget to visually present the data, or an an alternative widget from the default. *IMPORTANT!* The custom widget must already be provided in the codebase otherwise an error will occur. If this field is left blank (the default), the default widget for the specified datatype will be used, which should be good enough in 99% per cent of cases.

6.4.13 Derived Data Element (DDE)

IF a calculated field, a fragment of javascript outlined in *Derived Data Elements*. Leave blank if not a calculated field.

6.5 Datatypes

There are multiple data types, including:

6.5.1 String Datatype

Definition

A text value.

The maximum length of the string can be indicated.

Optionally a regular expression *pattern* can be entered in the pattern field of the cde definition

Examples

- hello world (a string containing a space)
- Mary (a word)
- ” ” (blank no quotes)
- The string 123
- ^^%%$^%$ff (non alphanumeric characters)

6.5.2 Integer Dataype

A whole number. Integer DEs can have a max or min value entered.

Examples

12, -1,0 etc.

6.5.3 Float Datatype

Definition

Real/Decimal numbers

Examples

- 3.1415
- 4.00
- -1.5
- 0.0

6.5.4 Alphanumeric Datatype

Definition

[A-Za-z0-9]*

Examples

- THX1138
- fred
- 234
- h4x0r

6.5.5 Boolean Datatype

Definition

Truth values

Examples

- True
- False

6.5.6 Range Datatype

For more sophisticated DEs, a DE can incorporate Permitted Value Groups (PVG).

Definition

A set of allowed values (usually represented as drop down list)

Ranges in RDRF are specified by the datatype keyword "range" and then selecting the appropriate *Permitted Value Group* This entails that permitted value groups be created first.

Examples

- shoe size : big, medium, small
- colour: red, blue , green

6.5.7 Calculated (Derived Data Element)

Definition

A value which is computed *client-side* from other values on the form.

To created a calculated DE enter "calculated" as the datatype and then fill in the calculation field of the DE.

Examples

A calculation (for BMI) could be coded as:

```
var height = parseFloat(context.CDEHeight);
var mass = parseFloat(context.CDEMass);
context.result = mass / ( height * height );
```

The "context" here is an abstraction representing the *other* cdes on the containing form. (Hence these other DEs must be present in some section of same form as the form containing the calculated field, else an error will result).

6.5.8 File Datatype

Definition

A file DE presents a file chooser widget to the user, allowing upload (and download) of a file from the user's local file system. NB. Only the uploaded file name is displayed - not the content.

Examples

A consent form field.

6.5.9 Date Datatype

Definition

A day, month, year combination

Examples

- 4th Jan 2008
- 8 Dec 2078

6.5.10 ComplexField Datatype

Definition

A DE used to aggregate other DEs horizontally on the page.

The intent is mainly stylistic

Example

- ComplexField(CDEName,CDEAge)

NB. This feature is experimental

6.6 Permissible Values and Permitted Value Groups

6.6.1 Permissible Values

A permissible value (PV) is just a single allowed value that comprises part of a value range.

6.6.2 Examples

If the associated value range was intended to be a size range then examples might be

- small
- medium
- large

In the GUI, individual permissible values will appear as selectable options in a drop down list.

6.6.3 Permitted Value Group

A permitted value group (PVG) is a set of permissible values. A PVG must be defined first, with PVs then defined and assigned to a PVG. PVGS are defined by navigating to "Permissible Value Groups" under "Settings".

Click "Add" to create a new PVG.

A value group has a code which must be a unique non blank string value.

Once a permitted value group has been created, add permissible values to it.

6.6.4 Examples

The decoupling of permitted value groups from cdes that make use of them allows different attributes

(e.g. shoe size , hat size) to share the same value ranges (E.g. small, medium, large) as is done in NINDS http://www.commondataelements.ninds.nih.gov)

6.7 How To Create a Registry in Ten Minutes

6.7.1 Modelling

1. Do this first on pen and paper!
2. Gather requirements of the data fields (*"DEs"*) required
3. For each data field required, decide its datatype. If a field is logically a *range*, work out the allowed *permitted values*. Depending on the datatype, decide any validation rules for a numeric or float data type (max and/or min), and for a string data type, the maximum length or pattern. Decide if any Derived Data Elements (calculated data type) are required.
4. Split them into logical groups (*sections*). Decide whether a section might be multiple
5. Portion related sections into *forms*
6. If a questionnaire is required for the registry, nominate a single form as a questionnaire

6.7.2 Creation

1. Login as an admin and navigate to "Registries" via the "Settings" button
2. Create a *registry* object and give it a name and code
3. Create any *permitted value groups* required (adding

any permitted values to the range.

4. Create DEs (one per data field) - OR note an existing DE that does the job (*IMPORTANT* deleting a DE

   may affect other registries which use this cde - in the current version there are no safeguards to prevent a used DE from being deleted. Proceed with caution!)

5. Create the section objects and enter the cde codes required into the elements field

   NB. This *MUST* be a comma-delimited list (E,g "CDEName, CDEAge" - no quotes)

6. Create forms and link them to the registry
7. Add section codes to the sections field

That's it as far as registry definition goes. The RDRF database now contains the definition of the registry. It is already usable by end users without any re-start - the defined forms are created entirely dynamically.

6.7.3 Registry Use

- To begin using the registry, login as a curator and assign patients to the registry.
- Patients can be added by navigating to the "Patient List" from the "Menu" button. Forms are then accessed for each Patient by clicking on "Modules".

6.7.4 Demo Site

- A demo site is up and running at: https://rdrf.ccgapps.com.au/demo/
- **Logins Provided (username/password):**
  - admin/admin (for definition of new registries)
  - curator/curator (for data entry)
  - clinical/clinical (for data entry on clinical forms)
  - genetic/genetic (for data entry on genetic forms)
- A Demo Contact Registry and Demo Clinical Registry for Myotonic Dystrophy are provided.

6.8 Importing and Exporting Registries

6.8.1 Importing

Importing a registry definition saved a yaml file. Login as an admin and navigate to "Importer" under "Settings", and select a previously exported registry yaml file from your file system.

6.8.2 Export

Exporting of a defined registry to a yaml file. Login as an admin and navigate to the registry required ("Registries" under "Settings"). Select the export custom action and run it.

6.9 Questionnaires

To fill out a questionnaire for registry with code fh (for example)

The member of the public navigates to the URL: <RDRF URL>/fh/questionnaire where RDRF URL is the site URL of the RDRF instance. (E,g, https://ccgapps.com.au/demo-rdrf/fh/questionnaire)

6.10 REST Interface

6.10.1 Updating CDE Data

Using curl: Posting a file:

curl -i -F value=@LPK.filvar.vcf <rdrf url>/<regcode>/patients/<patientid>/<formname>/<sectioncode>/<cdecode>

NB. The CDE with code <cdecode> must have datatype 'file'

Updating a non-file value

curl -i -H "Accept: application/json" -X POST -d <value> <rdrf url>/<regcode>/patients/<patientid>/<formname>/<sectioncode>/<cdeco

NB. This only works at the CDE level at the moment.

6.11 Retrieving Data

6.11.1 Examples

curl –request GET <rdrf url>/fh/patients/1/fh/fhBMI/CDEHeight

retrieves json of height (only)

curl –request GET <rdrf url>/fh/patients/1/fh/fhBMI

retrieves json of fhBMI section dictionary(code ==> value mapping)

curl –request GET <rdrf url>/fh/patients/1/fh

retrieves json of fh form (JSON dictionary of section dictionaries)

curl –request GET <rdrf url>/fh/patients/1

retrieves JSON of all forms in fh registry for patient 1.

6.12 Installation

We currently install RDRF on CentOS servers which is the supported platform. There is nothing preventing you to install RDRF on any other Linux distribution, but installing on CentOS is easiest using the RPMs we provide.

Theoretically, you could install on any platform that can run a Python WSGI application. Additionally you will need access to a database server (Postgresql), MongoDB, and Memcached.

6.12.1 How to install

In the *how to install* section we assume that you are going to install RDRF on CentOS using our provided RPMs.

Running installation with Docker

Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

For more information please visit the site https://www.docker.com.

Local deployment

Checkout the source code from the GitHub repository https://github.com/muccg/rdrf.

For stable and up-to-date code please use `master` branch:

```
# git clone https://github.com/muccg/rdrf.git
```

Enter the top directory of the source code:

```
# cd rdrf
```

To build all the necessary images run:

```
# ./develop.sh dev_build
```

After successful build run:

```
# docker-compose up
```

The application is available from:

```
# http://localhost:8000
# https://localhost:8443/app
```

or:

```
# http://<host_ip>:8000
# https://<host_ip>:8443/app
```

RPM Prerequisites

Extra CentOS repositories

The EPEL, IUS and CCG rpm repositories are needed to install packages that are not in the CentOS base repositories.

The way to add the EPEL and IUS repositories, depends on the version of CentOS you're having. The following example is for CentOS 6.6, for other versions of CentOS please follow the instructions at IUS Repos:

```
# yum install https://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# yum install https://dl.iuscommunity.org/pub/ius/stable/CentOS/6/x86_64/ius-release-1.0-14.ius.cento
# yum install http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-redhat94-9.4-1.noarch.rpm
```

To add the CCG repository:

```
# yum install http://repo.ccgapps.com.au/repo/ccg/centos/6/os/noarch/CentOS/RPMS/ccg-release-6-2.noar
```

MongoDB

RDRF uses MongoDB to store phenotype data. A simple MongoDB installation for evaluating RDRF can be achieved by installing the RPM:

```
# yum install mongodb-server
```

Start the MongoDB service with:

```
# service mongod start
```

RPM Installation of the RDRF web application under Apache

The IUS repository provides a `httpd24u` package that unfortunately conflicts with `httpd`. Therefore if you try to install `rdrf` and you don't have one of the `httpd` packages already installed you will get a conflict error. The recommended way (in the email announcing httpd24u) to get around this problem is to install the httpd package first and only after that install rdrf:

```
# yum install httpd mod_ssl
# yum install rdrf
```

This will add an Apache conf file to `/etc/httpd/conf.d` called `rdrf.ccg`. Please feel free to read through it and edit if required. When you are happy with the contents create a symbolic link for Apache to pick this config up automatically:

```
# pushd /etc/httpd/conf.d && ln -s rdrf.ccg rdrf.conf && popd
```

Database Setup

We assume that you have a database server (preferably Postgres) installed, that is accessible from the server you're performing the RDRF installation on.

Create the RDRF database

Please create a database (ex. `rdrf_prod`) that will be used by the RDRF application. We recommend creating a user called `rdrfapp` with no special privileges that will be the owner of the database.

Configure RDRF to use RDRF database

To change the database that RDRF points at you will need to alter the RDRF settings file. Open up `/etc/rdrf/rdrf.conf` in your editor and make sure the variables in the *database options* section (`dbserver`, `dbname`, `dbuser` etc.) are set correctly.

For more details see settings.

Initialise the RDRF database

The RDRF codebase employs South to manage schema and data migrations. To initialise the database:

```
# rdrf syncdb --noinput
# rdrf migrate
```

These will create the schema, insert setup data, and create initial users.

Caching and Sessions

By default RDRF uses database caching and file-based sessions but we recommend using memcached for both.

Changing both caching and session to memcached is therefore easy. Assuming you already have one or more memcached servers ready to go, all you need to do is open `/etc/rdrf/rdrf.conf` in your editor and set the `memcache` variable to a space-separated list of memcache servers.

This will make RDRF switch to memcached for both caching and sessions.

See:

- settings
- Django caching
- Django sessions

Last steps

At this stage we should have everything installed, the database, caching and sessions configured.

As a last step before starting the applications you should go through all the variables in `/etc/rdrf/rdrf.conf` and make sure everything is set to sensible values. See :ref:`settings`.

Restart apache

To start up the RDRF web application restart Apache:

```
# service httpd restart
```

Start MongoDB

To start up the MongoDB:

```
# service mongod start
```

Load fixtures

Create initial users and sample data elements (both files must be loaded in order):

```
# django-admin.py  load_fixture --file=rdrf.json
# django-admin.py   load_fixture --file=users.json
```

The following users/password are created:

- curator/curator
- genetic/genetic
- fhcurator/fhcurator
- clinical/clinical
- admin/admin

The sample registries are purely for demonstration purposes

NB. After designing a registry (along with its working groups) - users will need to be assigned to a working and registry before they will be able to see any data - this is accomplished by logging in as an admin and editing the given user.

At this stage you should be able to access the RDRF web application by browsing to https://YOURHOST/rdrf/.

6.13 Development Environment

6.13.1 Setting up your development environment

After downloading the source and decompressing:

1. first install docker (https://www.docker.com/) and python tools pip and virtualenv.
2. cd in to the source directory
3. issue the command: ./develop.sh start (this will install a tool which creates the application containers and wires them up.)
4. open a web browser at localhost:8000 and login with the (dev only!) admin account (password admin)
5. look at *How to create a registry*

6.14 Commands - Using develop.sh

6.14.1 usage

Usage ./develop.sh (pythonlint|jslint|start|rpmbuild|rpm_publish|unit_tests|seleniumlettuce|ci_staging)

6.14.2 start

./develop.sh start

This brings up the dev container - login at localhost:8000

Changes to the code are automatically picked up. Data and logs in ./data/dev

6.14.3 unit tests

./develop.sh unit_tests

6.14.4 selenium tests

./develop.sh selenium

6.15 Commands - Using fig

RDRF can also be run via fig on the command line inside the source directory (NB fig (http://www.fig.sh/) should be installed)

NB. cd into source folder first for these commands

6.15.1 Starting the development container

fig up

6.15.2 Running unit tests

fig -f fig-test.yml up

6.15.3 Running selenium tests

fig -f fig-selenium.yml up

6.16 Security

6.16.1 Application Level Security

RDRF is built on top of Django 1.8.4, the latest LTS (Long term support) release of Django. LTS releases get security and data loss fixes applied for a guaranteed period of time (3+ years.)

Django itself provides distinct levels of built-in security including:

1. SSL (Secure Socket Layer) security - all web traffic to and from application is encrypted.

2. CSRF (Cross-Site Request Forgery) checking: A method of insuring that man in the middle attacks (falsifying form submissions for example) are impossible.

3. Login restrictions of all "views"

4. In addition RDRF uses the Django Secure package (http://django-secure.readthedocs.org/en/v0.1.2/) middleware with all settings enabled by default.

5. RDRF itself includes a fully configurable permissions layer (role based security model) which restrict the visibility of forms (and fields) to specified user groups.

6. Furthermore, RDRF stores identifying patient contact/demographic data in a totally distinct database to any clinical/genetic data.

6.16.2 Notes on Operational Security

Any deployment of a registry will need to address operational security. This is security relating to the environment in which the software runs, and cannot be addressed by the software itself.

1. security of data on physical media

The registry framework stores data in PostgreSQL and MongoDB. The database these systems use should be encrypted. This ensures that data is protected if the storage hardware is (for example) stolen, reused, or returned to the manufacturer to address a fault.

In this sense, storage includes all physical media which is used to store registry data, including the volumes used by the database software, the volume on which the front end is installed, and any volumes used for operating system "swap" space.

2. inter-server / inter-datacentre encryption

Communication between the front-end and the databases should be encrypted. This is to guard against confidential data being intercepted "in transit". In addition to encryption, SSL certificates should be used (and databases and database clients configured to verify them) so that a third party cannot impersonate the database or a database client (known as a "person in the middle" attack.)

3. encrypted access to front-end

Access to the front-end should be via SSL configured web server. The SSL configuration should be modern (with processes in place to ensure it remains current) and have all security updates applied. This includes ensuring that a modern cipher suite is selected. Such a cipher suite will provide perfect forward security, reducing the consequences of a compromise of the server SSL key, while also allowing use of ciphers which increase client performance.

4. physical security - servers and infrastructure

The servers and related infrastructure should be maintained in a secure location, where the risk of unauthorised access, tampering or theft is reduced to a minimum. There should be documentation of who can access the facility, and when that access changes.

5. physical security - workstations

Workstations (including laptops) used to access the registry should require user authorisation, be subject to appropriate security policies, and have appropriate security software installed. On any workstation on which reports may be downloaded from the registry and stored, whole-disk encryption should be implemented on the device to guard against the risk of data exposure through theft or accidental loss.

C

D