Rigshospitalet

PET & Cyklotronenheden

Technical documentation and risk assessment for Tracershop

Christoffer Vilstrup Jensen

Resume

This document describes the current Tracershop ecosystem, a bookkeeping system for the production of radioactive tracers for clinical procedures produced at Rigshospitalet. The system was designed in 2004 and in the field of Computer Science such a system is considered ancient. This document presents a replacement system with the aims to improve the user experience, functionality, and improve IT-security, namely a react-django new web server and makes a risk assessment of the current and the new system.

Tracershop - Current system

The original Tracershop was originally produced in 2004. Various contributions to the ecosystem have been made over the years, however development the system stopped around 2012. The main functionality of Tracershop is fulfilling the documentations requirements in Rigshospitalets production of radioactive tracers specified by GMP. Rigshospitalet delivers radioactive tracers to many hospitals and scientific institutions, that would not be able to perform various PET and SPECT scans without the tracer. Therefore Tracershop should be considered a important piece of software.
The system is centered around a MySQL 5.1 database running on a openSUSE 11.2 distribution, which contains all the records and logs about the production of tracers. The user interface is a Zope 2-2.13 web interface allowing the users interface with the main database. It is normal that software packages deprecated after a period of support, and keeping a software product up to date is part of the maintenance of a software products life cycle, however that have not been possible due to insufficient resources.
An overview of the system can be seen in figure 1.

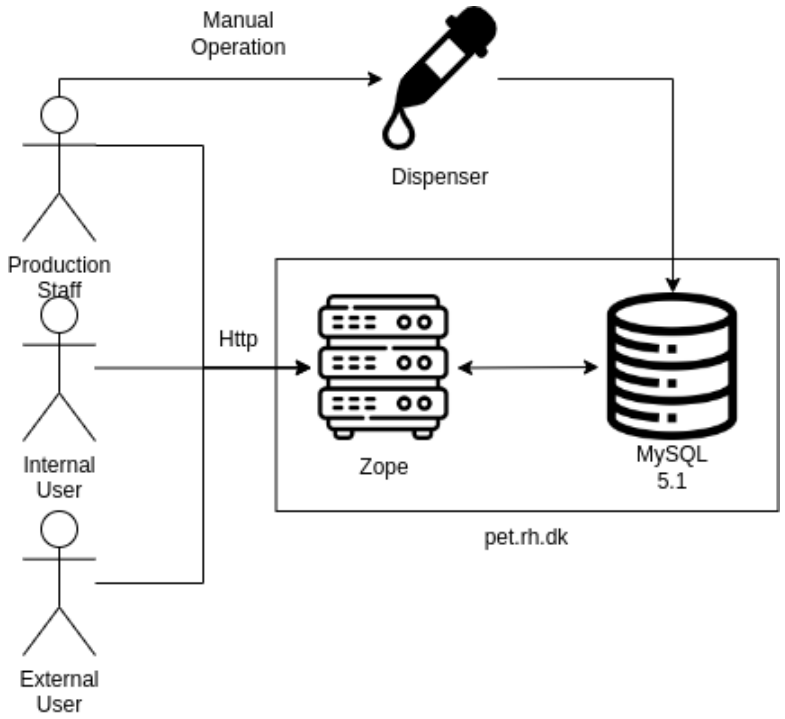

Figure 1: Tracershop system overview

There's a dispenser in the ecosystem. The dispenser ensures the dispensed amount of product and the digital record of the product is correct. The dispenser writes to it own database, which does not support multiple concurrent connections. A Tracershop component retrieves the data of the dispensed vial by opening the database as a file, seeks to an offset just before the new line was added and adds an entry in the database. The author of this script no longer works at Rigshospitalet. This solution is highly fragile, as an update to the database would render the script inoperable as the byte offset would no longer be correct. In such a case the script would read garbage would not be able to construct a valid entry and thus would mostly likely fail. At which point the production would have to manually write the data into Tracershops database, which introduces another human error.
Due to the nature of the dispenser, a less fragile solution is impossible, it's therefore recommended to replace the dispenser, with one fulfils the requirements of either:

- Writes dispensed tracer data to a database that allows multiple connections.
- Writes dispensed tracer data to a file.

Tracershop software components reached end of life, and as such continued usage is not advised. Tracershop was build with correctness in mind, however that's no longer sufficient for a good IT product. Sadly IT-criminality is on the rise, as such it necessary to bad actors, even in a low risk system, such as tracershop.
Currently it's the The Department of clinical physiology nuclear medicine which hosts the server.

Abstractions

Tracershop have two different types of orders. Activity based orders and Injection based orders. An "Activity Order" is defined as an order, where the user orders an amount of MBq radioactive tracer at a predetermined time slot known as a "deliver time". Hence it's the user responsibility to account for radioactive decay and order a sufficient amount tracer. While an injection order consists of a number of injections, and it's Tracershops responsibility to account for any decay between production time and injection time. Users er limited per user basis which injection tracers they can order.

Requirements

- A user can order radioactive tracer from the internet while not connected to the Capital Region intranet. A user can be limited in their selection of tracers and limited time window where such tracers are available for ordering.
- A user can review any order they have made, and view batch number of tracer that they have ordered. They cannot view or alter order, which doesn't belong to them.
- It's possible to track, which staff member produced which order. A released order must have a valid batch number.
- Non authenticated users cannot alter or view information in tracershop.

Database layout

The Tables in the Tracershop main database, note the tables casing does not conform to the snake casing that standard for SQL databases:

1. Log - Likely Zope related. No longer in use as the last entry was in 2010-03-18.
2. MiscData - Likely Zope related. Likely a temporary value container.
3. Roles - Zope related, defines different user roles in the Tracershop program.
4. Sessions - Likely Zope related, defines active user sessions.
5. Tokens - Likely Zope related, Likely defines the length of active sessions.
6. TracerCustomer - Defines which user have access to order which injection Tracer.
7. Tracers - Catalog of tracer available in tracershop.
8. UserRoles - Relates user to a Role from the Role table.
9. productions - Production of tracers.
10. storage - Old mails, assumed not in use.
11. t_orders - List of injection orders.

The database are not utilizing the foreign key restriction, meaning that the relations are ensured at application level and not the database level. It's recommend to utilize the these functionality to ensure that the database stays in a valid state. For instance if a tracer entry is dropped or updated, it may not be possible to determine the tracer of any historic order, that might have used this tracer.
Despite Activity orders having a tracer field, this is ignored by the web interface and assumed to be FDG.

New System

Tracershop is a system of individual components, so a simple switch of system is not advised, as any of new components would likely contain early adopter errors. While these errors likely are easy to resolve, they would still take the system down, which is unacceptable. The best plan of action would be replace each individual component, and if possible have old component running as a backup in the event that a new component has an error.
The largest and easiest to replace component is the web interface, as it can connect to the MySQL database and updates done by the new and old system can co-exists, as long as the old and the new system follows the same assumptions.
This report suggest a new web server. The project is open source found at:
https://github.com/demiguard/Tracershop
Once the basic functionality of a new component is fulfilled and the robustness of the system have been verified. The old system can be retried and any assumptions held by the old system can be dropped. The new system hosted on Center for IT and Medico technologies servers, who is responsible for general IT in the Capital Region, including maintaining the network spanning all the hospitals in the Region, and therefore would be in line with Rigshospitalet policies regarding software. To ensure the new system works correctly, the new system is using common software packages, these are:

- React - Frontend Javascript library developed by facebook. https://reactjs.org
- Django - Backend Python web server library https://djangoproject.com
- Channels - Django extension for using websockets. https://channels.readthedocs.io./en/stable

These packages are well supported and all in active development, so they will continue to provide security updates. The new Tracershop utilize websockets instead of http communication, because websockets allows the server to push updates to the users, because the protocol uses a persistance connection, while http does not use a persistent connection. This means that user always view up to date data, while in old system users might operate on out of date data, because another user have altered such data.

To ensure correct behavior of the new tracershop handles message in a three step plan seen below:

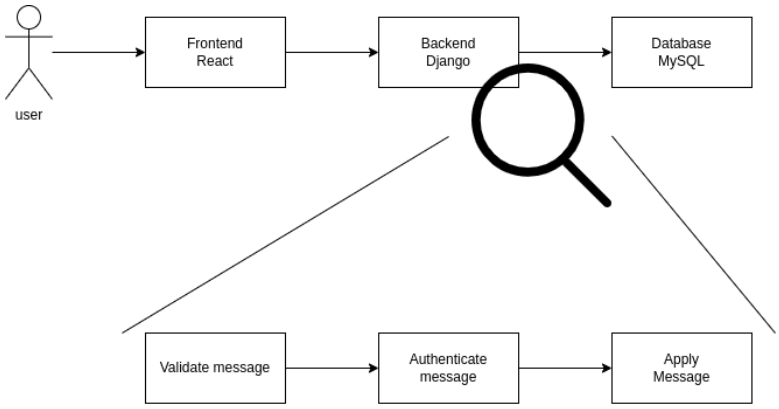

Figure 2: Message handling by the backend

1. Validate the message - ensure that the message is correctly formatted and therefore would not cause an error if processed by the webserver.
2. Authenticate the message - Now the server can assume, the message is of a valid structure and determine it type. Then it can check if the authenticated user is allowed to complete such a message.
3. Apply the message - Now the message is both valid and authorized, so the server will handle the message.

Since the frontend doesn't have any contact with the database, it is sufficient to ensure that the backend updates the database correctly on corresponding messages. This is done by a series of unit tests and end to end tests, which have a 92% code coverage, which proves the web server handles user message correctly.

Conclusion

The old tracershop fulfils the requirement set forwards by GMP, but is running on modules that have reached end of life. Therefore a new system is proposed, that fulfils GMP and no longer uses component, that is no longer supported.

Glossary

FDG [^{18}F]Fluorodeoxyglucose, a common radioactive tracer used for PET images. 2

GMP Good manufacturing practice, created by the EU and Danish medicines agency.. 1, 4