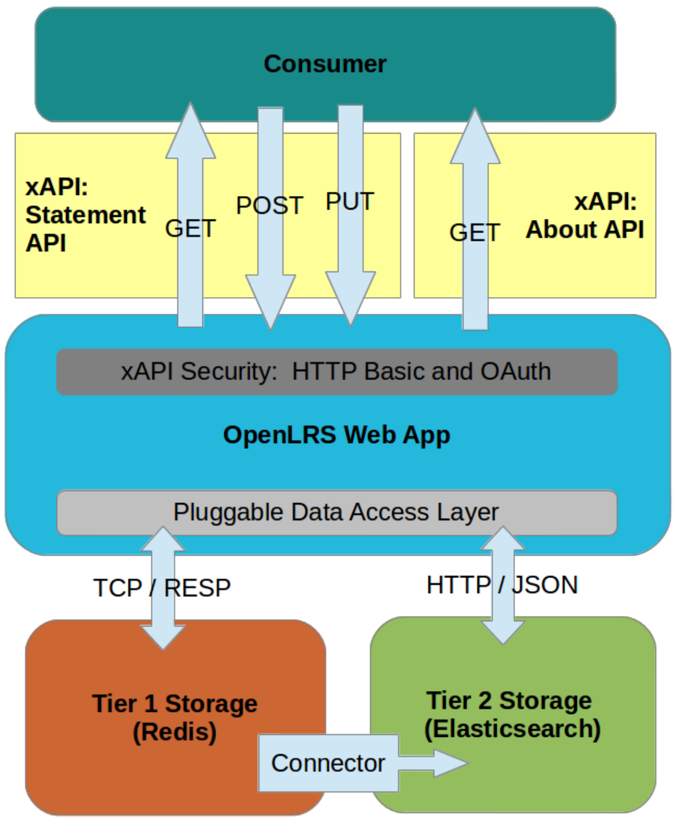
USUHS - OpenLRS Development - HLD

Overview

The OpenLRS web application will be a secure, standards-based, standalone Learning Record Store. It will be built on a scaleable architecture, using modern web technologies, and will provide fast reads and writes.

Diagram



Use Cases

- save a statement (POST)
- update a statement (PUT)
- retrieve a statement (GET)
- retrieve a group of statements (GET)
- retrieve "about" info (GET)

API

xAPI About

https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md#aboutresource

xAPI Statements

https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md#stmtapi

xAPI Security

- HTTP Basic Authentication
- OAuth 1.0a Authentication

https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md#security

OpenLRS Web Application

The OpensLRS web application provides the xAPI interface, secures the API, and manages the retrieval, storage, and indexing of learning records. A two-tier storage solution capability will be built-in, in order to allow the ability to leverage separate storage solutions, one that provides high speed writes and another that provides high speed reads. The web app will use a pluggable data access layer that allows flexibility in deciding which backend storage solutions to use. This is important because the response times and scalability will be heavily dependent on the storage and indexing solutions used. Security via xAPI HTTP Basic and OAuth 1.0a will be provided.

Data Access Layer

As mentioned previously, the data access layer will be pluggable, meaning that backend storage solutions can be swapped out as needed depending on cost/performance/infrastructure needs. The two-tier storage capability will allow for separate storage products to be used for "writes" (Tier 1) vs. "reads" (Tier 2), however, it will still be possible to use just a single storage product for both "reads" and "writes", if that is desired. For example, one could use MongoDB for Tier 1 and MySql for Tier 2, or just as well use Oracle DB for both Tier 1 and Tier 2.

Connector

If separate storage solutions are used for Tier 1 (writes) vs. Tier 2 (reads), then it may be necessary to use a "connector" that can transfer the data from Tier 1 to Tier 2.

Technology Stack

- Java 7
- Spring Boot

Tier 1 Storage

Key Critieria:

- high volume
- high speed writes
- transactional accuracy

The Tier 1 storage for the learning records will initially be Redis.

Redis Features:

- outstanding performance
- in-memory only (or persist to disk)
- master/slave async replication

http://redis.io/

Tier 2 Storage

Key Criteria

- high volume
- high speed reads
- indexing support

The Tier 2 storage used for the learning records will initially be Elasticsearch.

Elasticsearch Features:

- horizontally scaleable
- highly available

- very fast
- build on Lucene (for indexing)

http://www.elasticsearch.org/