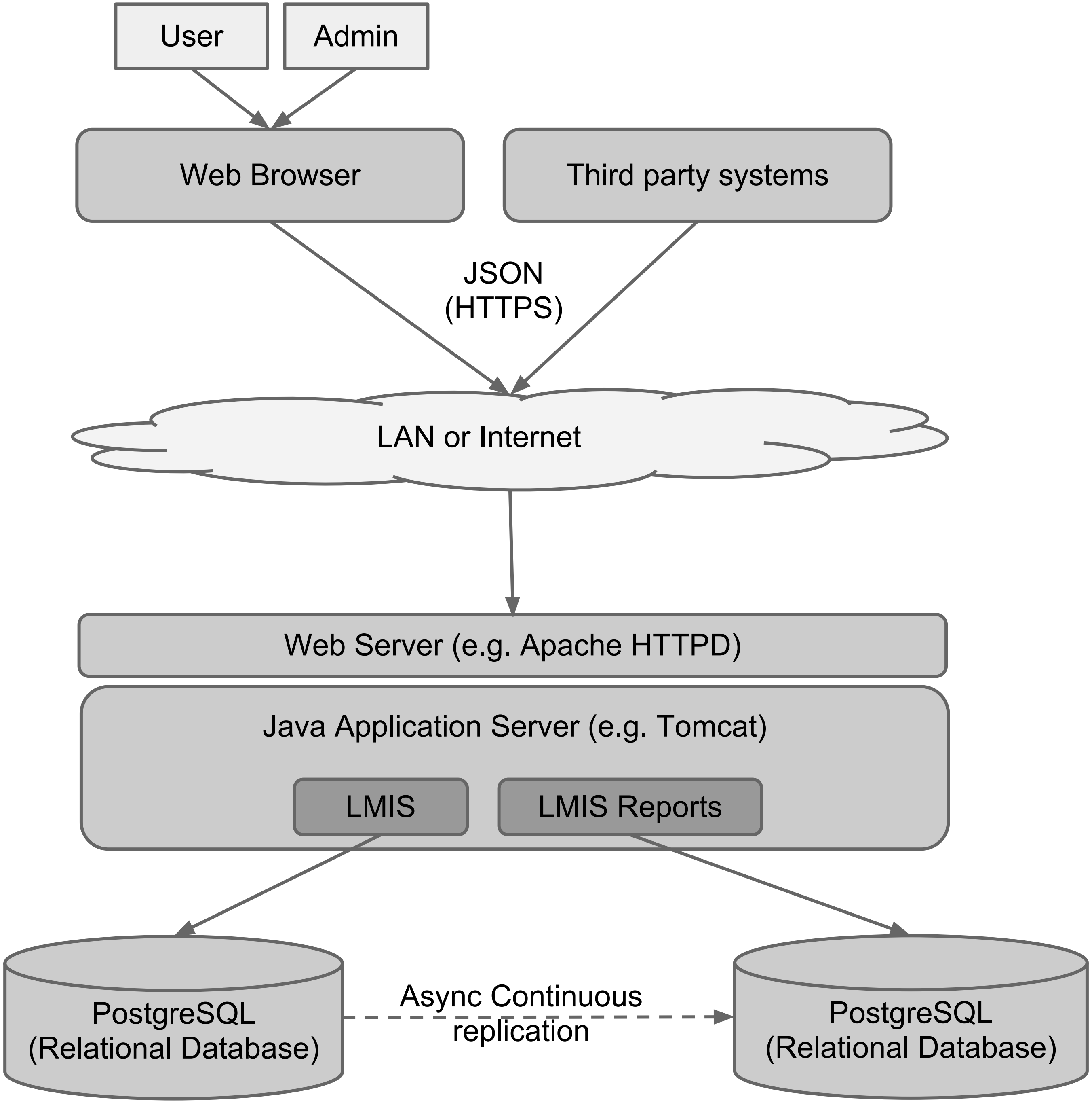
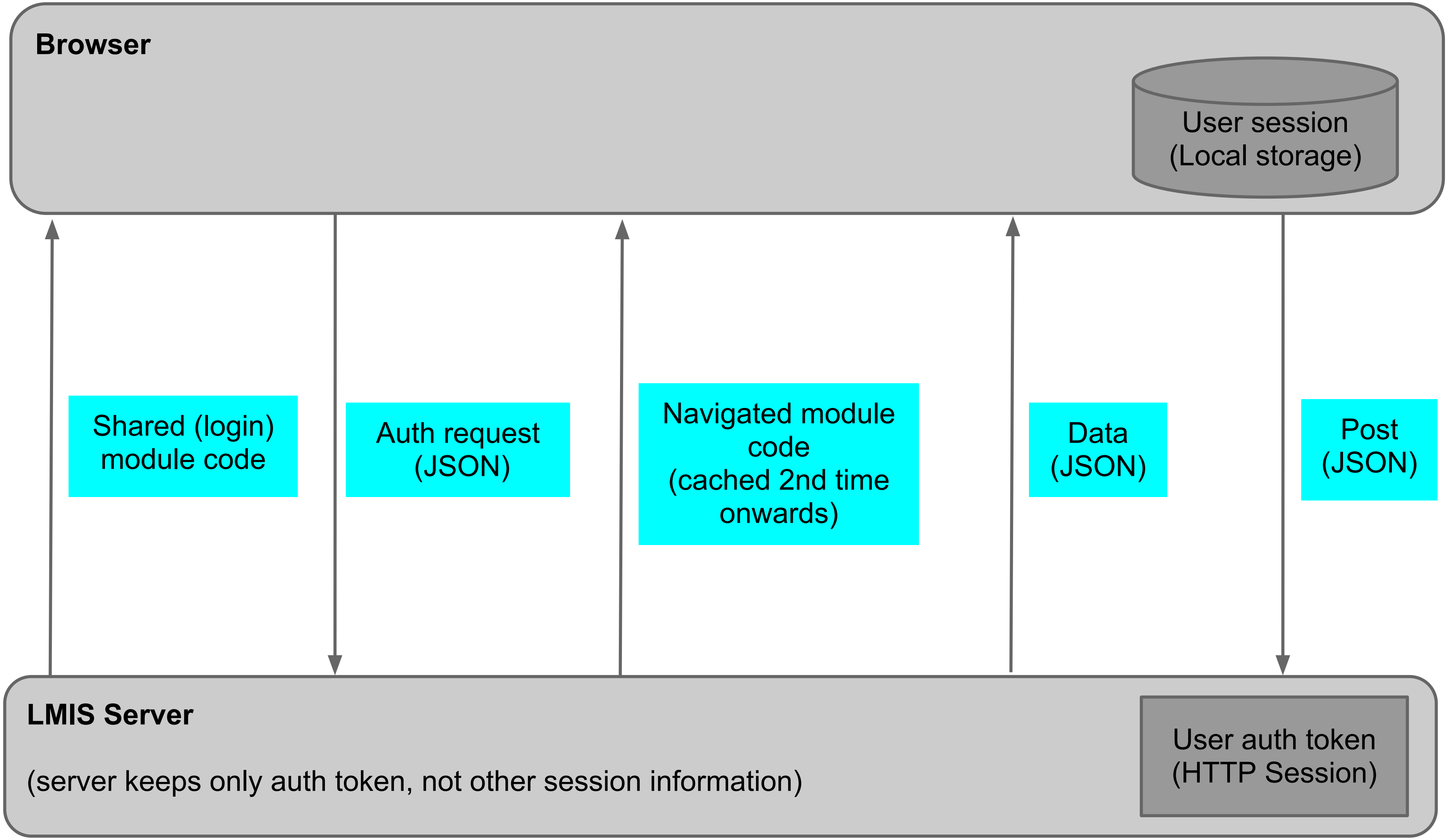
OpenLMIS Architecture


User
Admin
Web Browser
Third party systems
JSON
(HTTPS)
LAN or Internet
Web Server (e.g. Apache HTTPD)
Java Application Server (e.g. Tomcat)
LMIS
LMIS Reports
PostgreSQL
(Relational Database)
Async Continuous
replication
PostgreSQL
(Relational Database)

Presentation layer

Browser server interaction



Browser

User session
(Local storage)

Shared (login) module code

Auth request (JSON)

Navigated module code (cached 2nd time onwards)

Data (JSON)

Post (JSON)

LMIS Server

(server keeps only auth token, not other session information)

User auth token (HTTP Session)

Presentation layer (Web Browser)

Technology

HTML5, JavaScript, CSS

Tools/Libraries

AngularJS, jQuery & Bootstrap

LMIS modules

Facility, RnR, RnR-Template, Role, Schedule, Upload, Shared

- each module has all infrastructure layers - controller, model, routes

Presentation layer - Architecture considerations

HTML rendering in browser over server side

Use some system functionalities in offline mode

Local storage instead of cookies

Reduced upload traffic to server

IndexedDb or Local Storage (HTML5 tech for offline)

To be decided based on concrete requirements

Service layer

Service layer (LMIS)

Technology

Java, J2EE

Tools/Libraries

Spring framework, Spring MVC, MyBatis

LMIS modules

Authentication, Core, Requisition, Upload & Web

- Core has all domain abstractions for setup/reference data like Product, Program, Facility etc.

- each module has all infrastructure layers - service, domain, repository, etc.

Service layer (LMIS) - Architecture considerations

Spring

Dependency injection, transaction boundaries and REST

Unit test friendly

MyBatis over Hibernate *(for Object relational mapping)*

Explicit SQL easier to review for performance issues

Easier to learn

JSON over XML

Easier to map serialize to/from objects

Relatively easier for non-developers to understand

Reporting application

Platform

Jasper Reports (Java, J2EE)

Presentation in web browser

Out of box capabilities

User & access management

Excel,PDF,CSV,HTML and XML-over-HTTP

Development effort is mostly in defining SQL queries and customizing common templates

Database

Database

PostgreSQL over MySQL
Suitable for large and agile databases
Custom data types like tree-node (ltree), documents, etc.

No stored procedures & triggers (unless a must)
Languages supported by database not as powerful
Debugging, refactoring, testing is difficult
Database becomes bottleneck at large scale

Database server setup

Continuous replication over batch replication

Minimal loss of data on active node failure

Reports generated from passive instance in near real-time

Reporting from passive server

Non-performant queries not affecting transactional operations

Test for the correctness of backup process (side-effect)