Alma Courses REST API

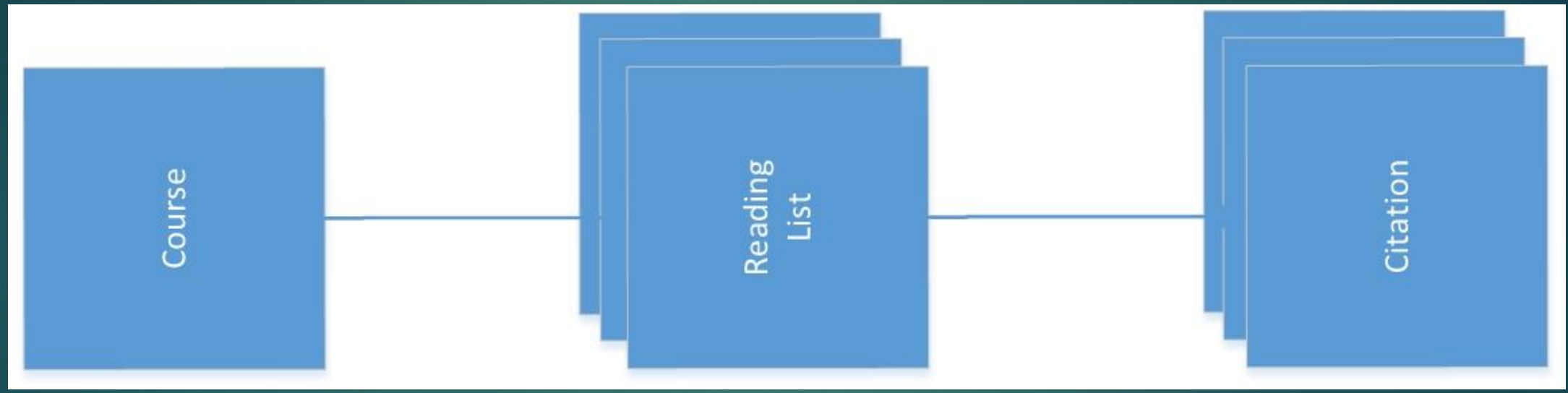
PERL CLIENT LIBRARY

CURTIN UNIVERSITY

IGELU 2017

ST. PETERSBURG

Alma Course Structure



What is REST?

A. REphrehensible SToryteller

B. REprobate STreet

C. RECreational STaffroom

D. REpresentational State Transfer

E. One of the above

Alma Courses REST API Overview

- Basic Course “things” (pre-Leganto)
  - Course, Reading List, Citation
- Other Course “things” (Leganto and on)
  - Reading List Owner, Citation Tag
- Basic REST operations (CRUD)
  - C(reate
  - R(ead
  - U(pdate
  - D(elete

Why use the Alma Courses API?

- Alma Course Loader restriction
  - Any time but only once
- Flexibility
  - Run once a day using Alma Course Loader
  - Run twice or more a day using Courses API
- Failed job runs
  - Points of failure – network, database, Alma
  - Retry manually at will

PERL Client Library

- Approach
  - Work with objects, not REST APIs
  - Courses, Reading Lists, Citations
- Abstracts and re-frames the Courses REST API
  - Hides basic REST API logic
  - Simplified functions
    - Create courses, reading lists, citations
    - List courses, Read next course, reading list, citations, etc
    - Update course, reading list, citation
    - Delete course, reading list, citaiton

REST? CRUD? MLWD? What??

- REST and CRUD
- MLWD
  - Holder for function-objects
  - Each course object has an associated function object
    - Course, Reading-List, Citation
    - Implements the REST function
    - Handles errors, parsing, sequencing, etc

REST Function	MLWD Function Object
Create	**M**aker
Read	**L**ister
Update	**W**riter
Delete	**D**estroyer

An Example

- List all courses
  - ACTIVE
  - UCPD processing department
- Load each course completely
  - Course
  - Reading lists
  - Citations
- Process each course + reading list(s) + citations

Perl Client Library – Pre-amble

```
my $rconf = {
     api_key => $apikey,
     api_base => "/almaws/v1",
     host => "https://api-ap.hosted.exlibrisgroup.com",
     timeout => 10,
     tries => 3,
     rest_debug => 1,
     format => 'xml'
};
my $client = Alma2::REST::Client->new($rest_conf);

my $crud = Alma2::Reserves::CRUD->new($client);     # the CRUD holding object
my $clister = $crud->lister()->course();            # the course lister
my $rlister = $crud->lister()->reading_lister();    # the reading-list lister
my $nlister = $crud->lister()->citation();          # the citation lister
```

Perl Client Library – Prepare, invoke

```
# preparing arguments for listing
my $pdi = 'UCPD';     # University Course Provisioning Department
$clister->setOptions(chunk => 50, offset => 0});

# call the listing function
my $rc = $clister->listAll();
die Dumper($client->print_error()) if !rc;
```

Perl Client Library – Main Loop

```
while ( my $course = $clister->next() ) {
    my $status = $course->get('course.status');
    next if $status =~ /INACTIVE/;
    my $pd = $course->get('course.processing_department.value');
    next if $pd !~ /$pdi/;

    load_rlists($course);

    # process / dump / do whatever you want to with the course
    ....
}

exit;
```

Perl Client Library - Implementation

```
sub load_rlists {
      my ($course) = @_;
      my $course_id = $course->getText("course.id");
      $rlister->listAll($course_id);                          # HTTP GET /courses/$course_id/reading-lists
      while (my $rlist = $rlister->next()) {
            $course->addReadingList($rlist);
            load_citations($course_id, $rlist);
      }
}

sub load_citations {
      my ($course_id, $rlist) = @_;
      my $rlid = $rlist->getText("reading_list.id");
      $nlister->listAll($course_id, $rlid);                   # HTTP GET /courses/$course_id/reading-lists/$rlid
      while (my $citation = $nlister->next()) {
            $rlist->addCitation($citation);
      }
}
```

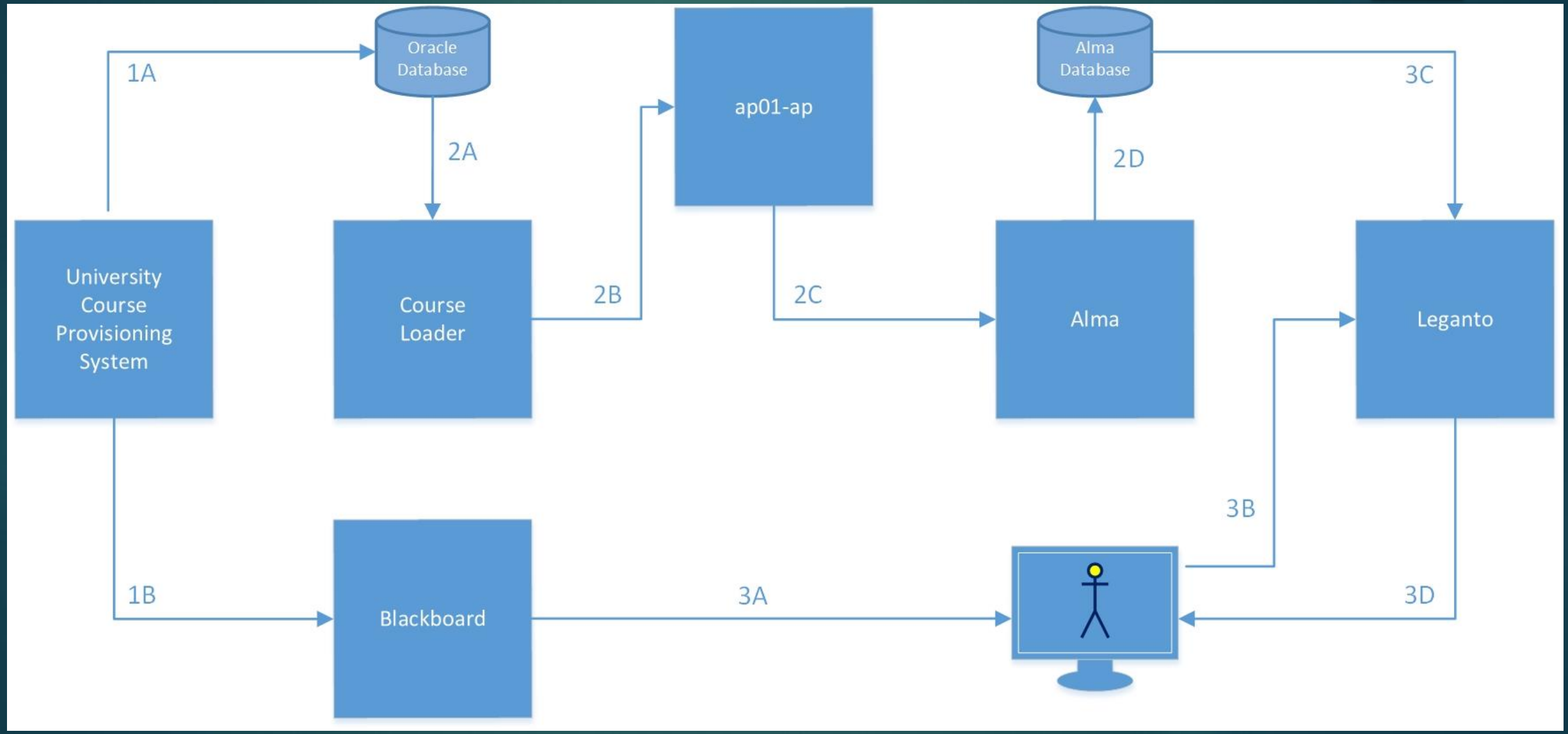
Comparing REST and Alma2::Reserves

Raw REST	Alma2::Reserves
Deal with REST calls directly	Conceals REST calls
Deal with errors (parsing,handling,etc)	Conceals 90% of REST interaction
Parse in XML or JSON	Does this invisibly for you
Complexity and errors	Simplifies programming. Fewer errors
Attention to lots of repetitive detail	Lets you focus on business logic

Applications

- Daily Course Loader
  - Source Online Course provisioning system
  - Create courses, assign ownership
  - Update course titles and owners
- Periodic Course Cleaner
  - Deactivate old courses by date and processing department
- Periodic Mailer
  - Mix SQL database, Analytics API, Reserves API, Curtin internal data sources
  - List only staff who are academics and own Courses

Integration – Course Loader



QUESTIONS
&
ANSWERS