lyk.io

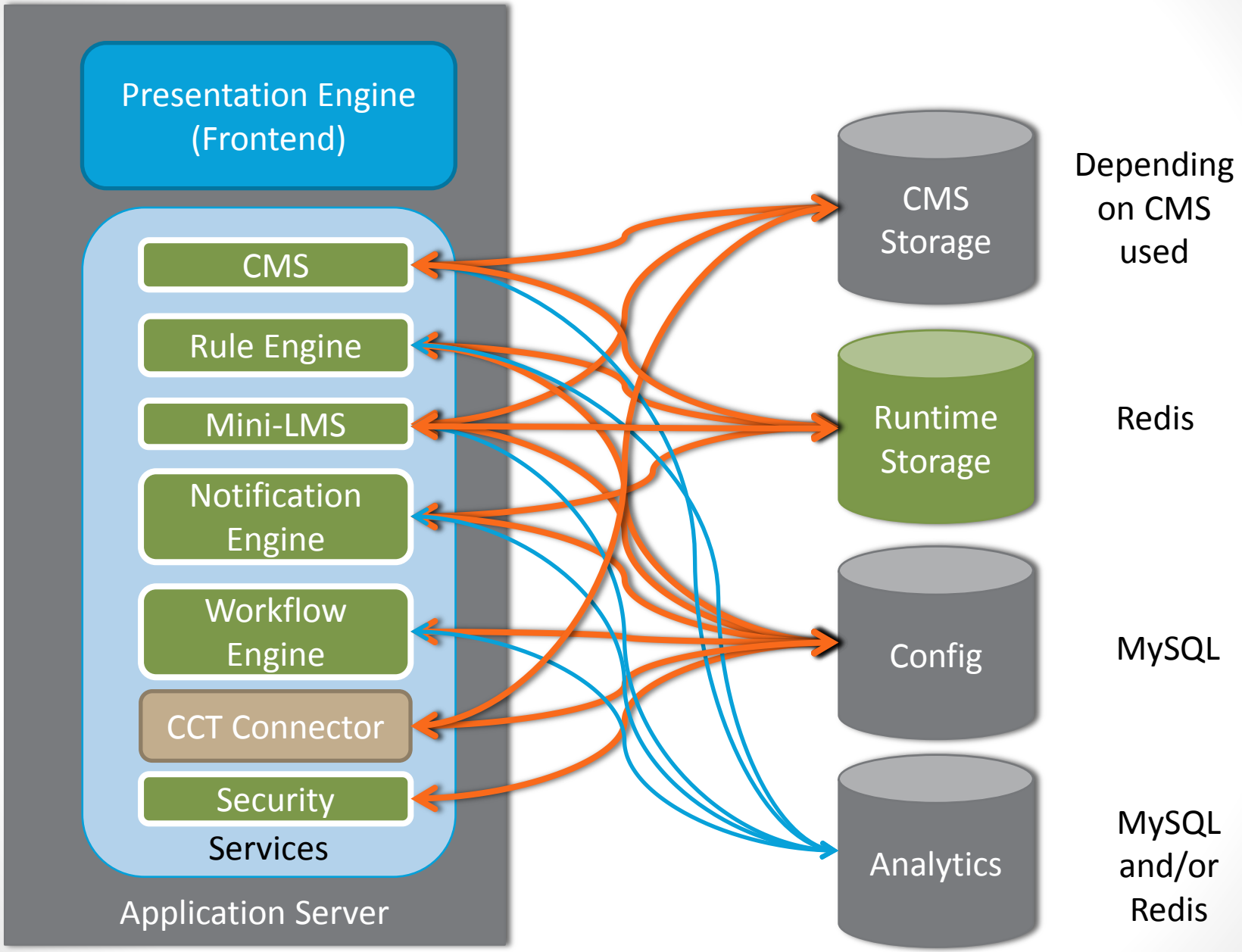
Components / platform / architecture

Agenda

- Overview - High level platform
- Technologies to be used
- Event based / triggers
- Multi-tenant architecture and deployment strategies

- Engineering Structure, now and evolution to ‘15

High level components diagram



Technologies

- Frontend technologies:
  - Play Framework
    - Play is a reactive and highly scalable web framework that optimizes development workflow without compromises in capability or development velocity
  - Optionally: AngularJS
    - AngularJS is a Google framework that allows expanding the HTML5 vocabulary to allow dynamic content
- Backend technologies:
  - Spring Framework
    - Spring is a very popular Java framework and lOC container that provides comprehensive support for developing enterprise class java applications
  - Hibernate
    - Hibernate is an Object – Relational Mapping library for Java, creating an abstraction towards any RDBMS and handling persistence in a uniform and optimal way

Technologies - Content

- Content – CMS functionality
  - Develop a custom-tailored solution to support simple CMS needs (author – publish – versioning – sandboxes)
  - Alternative: CMIS - an OASIS specification (Open Standard) backed by industry leaders including Afresco / HP Autonomy / EMC Documentum et al. Literally a “bazooka to kill a fly” solution; opens up a series of possible implementations including open source or commercial vendors (may utilize a clients’ existing installation).

Technologies - LMS

- Learning Management System
  - A lightweight in-house solution to cover needs of the platform
  - Possible implementations of client (integration)
  - Using moodle or similar out-of-the-box maybe overkill at this point

Technologies – Database

- Persistence layer – Hibernate
  - Using Hibernate for persistence provides freedom to choose the underlying database depending on client's needs. Possible candidates:
    - MySQL – free (commercial support from Oracle)
    - Oracle RDBMS – commercial
    - PostgreSQL – free (commercial support from EnterpriseDB)
  - Persistence may be selected at time of implementation where specific optimizations may be applied depending on choice of DB.

Technologies – Events & Rules

- Realtime Analytics via event processing:
  - REDIS provides:
    - Performance: in-memory operations (persistence is also available)
    - Efficiency: Optimized for storage of lists, hash-sets, bitmaps, etc
    - Scalability: single-thread model, atomic operations
- Events distribution via pub-sub model:
  - Hazelcast provides:
    - Publish-Subscribe (topics) out of the box
    - A data-grid of distributed memory with cluster elasticity
    - Resilience towards failures: nodes can fail without data loss
    - Low-footprint: small library

Technologies – Workflow

- Using workflows from the start
  - Session is a workflow for groups
  - Course is a workflow
  - Onboarding is a workflow, customizable by the implementor
  - Author – Review – Publish is a workflow
- Solutions – in-house state machine or workflow engine
  - State machine
    - Can be simple (needs development)
    - May need serious re-engineering for advanced flows
  - Workflow engine
    - Activiti – Open source, lightweight, BPMN2
    - jBPM – Open source, heavier, more mature

Events / Counters

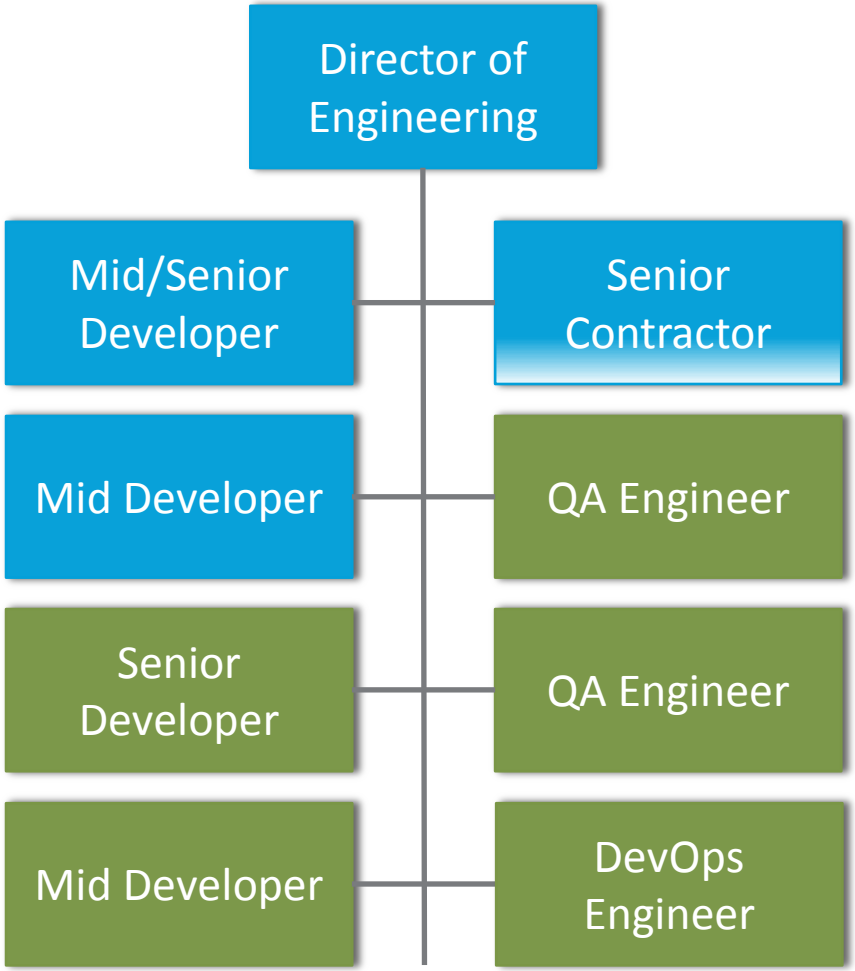
- System and User actions generate events
- Events are propagated through the system
- Components register to receive events
  - Listeners perform operations on events (+comment, +review)
    - Aggregate / Group (#comments per day/month/year)
    - Count (total comments / total reviewed / contributions)
    - Sum (over aggregates or on its own)
  - Data stored in time-series with granularity per need
- Rule engine monitors events and generates further events (e.g. Level up, Award, Certificate)
- Rule engine has default configuration but is configurable also

Multi-tenant from the start

- Deployment topology in-premises (deployment in containers)
  - A single tenant, using dedicated VMs
    - RDBMS server(s) – for configuration storage
    - CMS server(s) – for content and data streams (e.g. video)
    - REDIS server(s) – for runtime data
    - Application server(s) – for business logic / hazelcast
- Deployment topology in cloud (AWS / Azure / Google Cloud)
  - Multi-tenant, hosting multiple organizations
    - RDBMS server(s) – multi-tenant deployment for conf storage
    - CMS server(s) – multi-tenant for content and data streams
    - REDIS server(s) – multi-tenant for runtime data
    - Application server(s) – multi-tenant / single-tenant for business logic / hazelcast - *ease of configuration per implementation*

Questions

Engineering – lyk.io – CCS



Q4/14

Q2/15