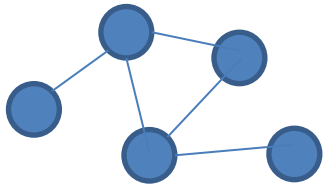
Technik – Systemtests – Entwicklung

Alfons Seelen, hbz
Julian Ladisch, VZG

FOLIO-Infotag Göttingen, 26. April 2018

Technisches Konzept

- Offene Plattform: Library Service Platform (LSP)
- Plattform stellt Infrastruktur für funktionale Module bereit
- Funktionale Module ⇨ eigenständige Programme
  - Können unabhängig voneinander entwickelt werden
  - Können einzeln ausgewählt und installiert werden
  - Kommunikation über Schnittstellen
- Design orientiert sich an Microservice-Idee



Technisches Konzept

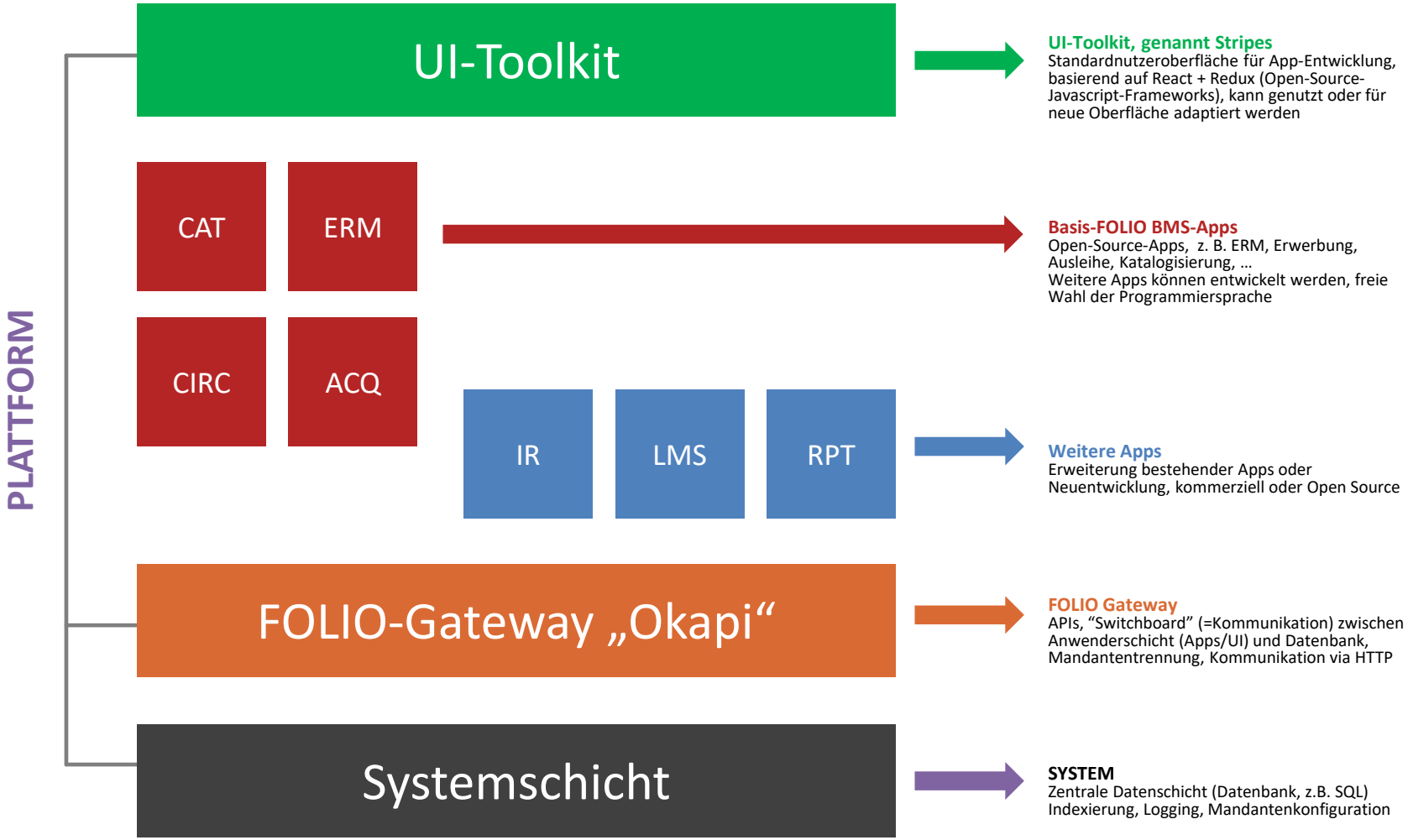
- Unterstützung verschiedener Support-Modelle
  - cloud-basiert, Hosting, lokal
  - kommerziell, Verbund, selber
- Mandantenfähig
- Flexibel erweiterbar, modular
- „Plug and Play“-Applikation
- Basierend auf heutigen Anforderungen mit Ausrichtung auf zukünftige Bedürfnisse

Plattformdesign

„Durchgängig APIs“

- Das bedeutet, dass
  - jeder Entwickler mit jeder Schicht in der Plattform interagieren kann, und
  - keine Komponente zu groß ist, um sie zu ersetzen.

Technologien



Technologien

Moderner Softwarestack
aus bewährten Komponenten

Frontend (= im Browser)

- JavaScript (ECMAScript 6)
- React/Redux

Backend (= auf Server)

- Java 8
- Vert.x (asynchrone Kommunikation)
- RAML
- PostgreSQL
  - JSONB (NoSQL) und
  - relationales SQL

React und Redux

- Sind Open-Source-JavaScript-Webframeworks für Single-Page-Applications (SPAs) = Einseitenwebanwendungen
- React bietet ein Grundgerüst für die Ausgabe von User-Interface-Komponenten in HTML
- Redux ist ein Datencontainer, vereinfacht Lesen vom und Schreiben zum Backend

- https://reactjs.org/ und https://redux.js.org/

Stripes

- Javascript-Programmbibliothek für Frontendmodule
- Basiert auf React + Redux
- Zugeschnitten auf Okapi
  - Kommunikation via Okapi zu Backendmodulen
  - Granulare Nutzerrechte
  - Locale (Sprache, Datumsformat, …)
  - Hotkeys (Tastaturabkürzungen)
  - Logging via Okapi
- https://github.com/folio-org/stripes-core/#readme

vert.x

- Bibliothek für Java
- Ermöglicht einfache Nebenläufigkeit
  - Umgeht viele Probleme paralleler Programmierung
  - Asynchrone Kommunikation
  - Reaktive Programmierung
  - Entwurfsmuster „Reactor“
- http://vertx.io/

RAML

- RAML = RESTful API Modeling Language
- Schnittstellenbeschreibung der Module
- Daraus wird automatisch erzeugt:
  - Schnittstellendokumentation, siehe https://dev.folio.org/doc/api/
  - Java-Code (Interfaces)
  - Validierung, die Okapi beim Schnittstellenaufruf durchführt:
    - Erforderliche Benutzerrechte vorhanden?
    - Datenformat korrekt?
- https://github.com/folio-org/raml-module-builder

Datenbankauswahl

- PostgreSQL
  - 2016 Tests auch mit MongoDB
- Wahl fiel auf PostgreSQL. Grund: unterstützt gleichzeitig (!)
  - sowohl relationales SQL-Datenbankmodell
  - als auch dokumentenbasiertes NoSQL-Datenbankmodell
- NoSQL = Not-only-SQL, in diesem Fall dokumentenbasiert (JSON-Dokumente)
- PostgreSQL kann JSON-Dokumente als JSONB verarbeiten, also in einem effizienten binären Format: Das JSON-Dokument wird in seine Bestandteile zerlegt und dadurch indexierbar gemacht.
- FOLIO speichert die meisten Daten als JSONB.

Mandantentrennung

- Mandant = völlig unabhängige Institution (Institutsbibliothek kein Mandant, sondern hat granulare hierarchische Zugriffsrechte)
- Je Mandant eine eigene logische Datenbank (PostgreSQL „Schema“) mit eigenem Datenbanknutzer (PostgreSQL „Role“)
- Datenbankverbindung mit Datenbanknutzer, PostgreSQL garantiert Mandantentrennung

Datenbankbetrieb

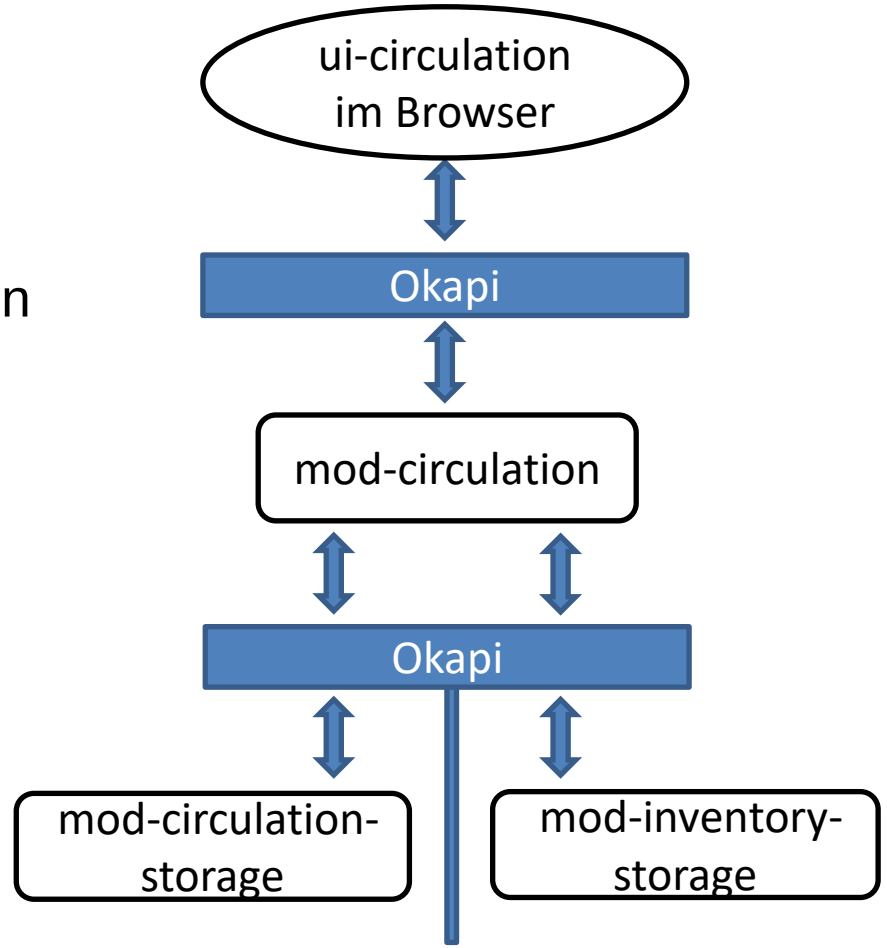
- Jedes Storage-Modul kann eine eigene PostgreSQL-Instanz starten.
  - Nützlich für die Softwareentwicklung.
- Per Parameter kann gemeinsame externe PostgreSQL-Installation angebunden werden.
  - Genutzt bei Demo- und Test-Installationen
  - Ermöglicht Hochverfügbarkeit und Replikation durch PostgreSQL-Cluster

Intermodulkommunikation

Beispiel Ausleih-App
mit drei Ausleih-Modulen:

Zu einem Medium kombiniert mod-circulation die Ausleihdaten (mod-circulation-storage) und Titeldaten (mod-inventory-storage).

*-storage = Datenbank-abstraktionsschicht



Technische Evaluation

- Die technische Basis der FOLIO-Plattform haben sowohl Vertreter der OLE-Community als auch EBSCO evaluiert.
- Ergebnis: Keine grundsätzlichen Bedenken, FOLIO ist auf dem richtigen Weg.
- Verbesserungsvorschläge/Hinweise wurden inzwischen größtenteils umgesetzt.

Wie wir arbeiten

- Verschiedene Teams, verschiedene Organisationen, verschiedene Länder
- Product-Owner sind regelmäßig anwesend in den wöchentlichen Entwickler-Meetings
- Tickets/Aufgaben werden in diesen Meetings besprochen

Code-Qualität

- Nachhaltiger Code ist uns wichtig
- Vorteile bei langfristigen Projekten:
  - Weniger Bugs/Fehler durch Testing
  - Lesbarkeit: einfacher zu bearbeiten

Continuous Integration

Schritte, die neuer Code durchlaufen sollte:

- Der Code wird automatisiert überprüft (ESLint/SonarLint/SonarQube)
- Automatisierte Tests werden ausgeführt
- Entwickler machen Code Reviews

Wenn Code eingebaut:

- Abnahme durch Product Owner/SIGs

Softwareentwicklung

http://dev.folio.org/

- Einstiegsseite für Entwickler
- Zugang zum Code
- Richtlinien
- Dokumentation
- Tutorial/Curriculum
- Komponenten und Mitarbeiter

Systemtest

- hbz und VZG betreiben mehrere FOLIO-Testinstallationen
- Eigene umfassende Systemtests
- Schwerpunkt: deutschland- und verbundspezifische Anforderungen

- Weitere Systemtest durch die internationale Community
- Beispiel: Test der User Management App durch Bibliothekare

Barrierefreiheit – Vorgaben

- Accessibility (= Barrierefreiheit) SIG hat festgelegt:
  - ➢ FOLIO muss WCAG 2.0 Stufe AA erfüllen
- WCAG = Web Content Accessibility Guidelines
  ist ein internationaler Standard: ISO/IEC 40500:2012
- WCAG 2.0 Stufe AA in der EU gesetzlich verpflichtend für alle staatlichen Einrichtungen ab 23. September 2019

Barrierefreiheit – Prinzipien

- WCAG-Prinzipien:
  - Leicht wahrnehmbar
  - Leicht bedienbar
  - Gut verständlich
  - Robust
- Dadurch auch für Menschen mit körperlichen oder geistigen Einschränkungen nutzbar
- Benutzerfreundlichkeit hilft aber auch allen anderen
- „Einfach für alle“
- WCAG enthält konkrete, überprüfbare Kriterien, Checklisten
- https://www.w3.org/Translations/WCAG20-de/

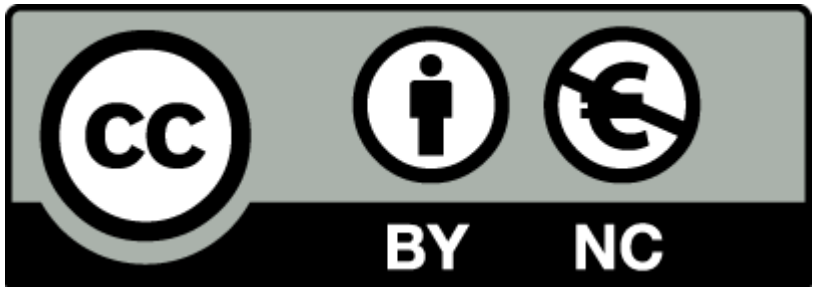
Barrierefreiheit – Test

- Accessibility SIG hat FOLIO einem Test unterzogen
- VZG hat dafür https://folio-demo.gbv.de bereitgestellt
- Einzelne gefundene Probleme, die derzeit behoben werden, u.a.:
  - unklare Beschriftungen und Icons
  - unterschiedliche Bezeichnungen für dieselbe Sache
  - fehlende oder doppelte Beschriftung (Screenreader)
  - zu geringer Kontrast
  - Bedienung per Tastatur (Tab, Pos1, …) unvollständig
  - Bedienung per Sprache unvollständig
  - funktionslose Button nicht ausgegraut
  - Suchfunktionalität arbeitet in jeder App anders
  - farbkodierte Information für Farbenblinde nicht erkennbar
- FOLIO ist ganz überwiegend barrierefrei dank Stripes/React

Vielen Dank!

Alfons Seelen
alfons.seelen@hbz-nrw.de

Julian Ladisch
julian.ladisch@gbv.de



Maßgeblich für diese Präsentation ist das gesprochene Wort.