FORMCENTRIC

Installationshandbuch

Version 4.5.2

Formcentric Analytics: Installationshandbuch

Breite Str. 61, 22767 Hamburg
Deutschland

Einschränkung der Gewährleistung

Inhaltliche Änderungen des Handbuchs und der Software behalten wir uns ohne Ankündigung vor. Es wird keine Haftung für die Richtigkeit des Inhalts des Handbuchs oder Schäden, die sich aus dem Gebrauch der Software ergeben, übernommen.

Warenzeichen

Innerhalb dieses Handbuchs wird auf Warenzeichen Bezug genommen, die nicht explizit als solche ausgewiesen sind. Aus dem Fehlen einer Kennzeichnung kann nicht geschlossen werden, dass ein Name frei von Rechten Dritter ist.

Beachten Sie bitte: Ausgedruckte Exemplare unterliegen nicht dem Änderungsdienst.

Zugang zu Dokumentationen

Sie finden immer die aktuelle Version des Installationshandbuchs im Hilfecenter unter help.formcentric.com. Ältere Versionen und weiterführende Informationen stehen Ihnen im Formcentric Helpdesk unter helpdesk.formcentric.com zur Verfügung.

1. Einleitung

Dieses Handbuch beschreibt die Installation der Formcentric Analytics Backend- und Reporting-Webapp.

Zu Beginn werden die Schritte zur Installation der Formcentric Analytics Backend-Webapp beschrieben, gefolgt von den entsprechenden Anweisungen für die Formcentric Analytics Reporting-Webapp. Dabei erfahren Sie, welche Property-Dateien anzupassen sind, um die Webanwendungen gemäß Ihren Anforderungen zu konfigurieren. Sowohl die Backend- als auch die Reporting-Webanwendungen werden Ihnen sowohl als WAR-Datei als auch als Spring-Boot-JAR zur Verfügung gestellt. Wir empfehlen die Verwendung der Spring-Boot-Variante.

Die folgende Grafik veranschaulicht das Gesamtsystem von Formcentric Analytics und bietet einen ersten Einblick in die Architektur.

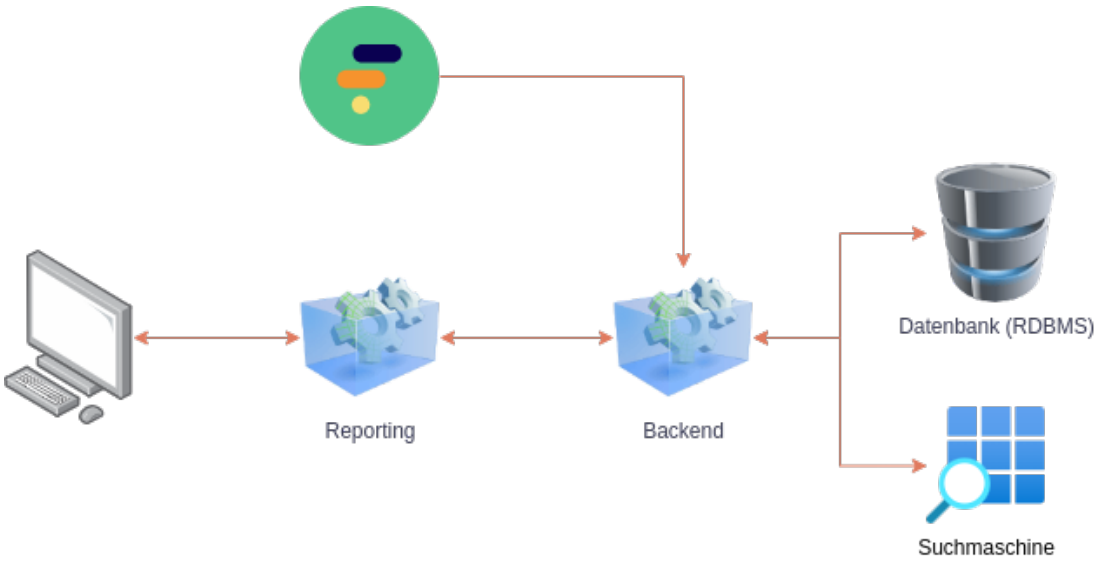

Abbildung 1.1. Analytics Komponenten

Im Folgenden finden Sie eine kurze Beschreibung der einzelnen Komponenten:

Reporting: Das Reporting stellt die Benutzeroberfläche bereit, die Sie in Ihrem Browser sehen. Mit dieser Anwendung können Sie Analytics verwalten und die gesammelten Daten einsehen.

Backend: Das Backend liefert die Daten aus der Datenbank (RDBMS) und der Suchmaschine an das Reporting. Außerdem nimmt es Daten von den abgesendeten Formularen von Formcentric entgegen und speichert sie.

Datenbank: Die Datenbank dient zur Speicherung der abgesendeten Formulare. Folgende Datenbankmanagementsysteme (RDBMS) werden unterstützt: PostgreSQL, MySql, MariaDB, Oracle, Microsoft SQL Server und DB2.

Suchmaschine: Für sämtliche Suchvorgänge wird entweder *Solr* oder *Elasticsearch* verwendet. Die Suchmaschinen ermöglichen eine schnelle und effiziente Durchsuchung der Daten in Analytics.

2. Backend-Webapp

Erfahren Sie in diesem Kapitel alles Wissenswerte über die Einrichtung, Installation und Verwaltung der Formcentric Analytics Backend-Webapp. Es führt Sie durch die notwendigen Voraussetzungen für den Betrieb und bietet Ihnen verschiedene Möglichkeiten zur Installation. Darüber hinaus erhalten Sie Einblicke in das Monitoring und Management der Anwendung sowie detaillierte Erklärungen zu den Konfigurationsparametern sowohl für Formcentric Analytics als auch für die Solr Enterprise-Suchmaschine. Zusätzlich behandelt dieses Kapitel den Zugriff aus dem Content-Management-System sowie die erforderlichen Zugriffsrechte für den Datenbank-Benutzer.

2.1. Voraussetzungen

Dieser Abschnitt beschreibt die Voraussetzungen für den Betrieb der Formcentric Analytics Backend-Webapp. Vor der initialen Installation oder einem Update von Formcentric Analytics, können Sie die unterstützten Versionen der Drittkomponenten den folgenden Tabellen entnehmen.

Nicht alle Komponenten sind für den Betrieb zwingend erforderlich. So bietet beispielsweise die RabbitMQ-Anbindung eine optionale Integrationsschnittstelle, die bei Bedarf jederzeit hinzugefügt werden kann. Weitere Informationen erhalten Sie in den entsprechenden Abschnitten.

2.1.1. Datenbanken

Formcentric Analytics unterstützt die folgenden Datenbanken.

Database	**Status**
PostgreSQL	
PostgreSQL 17	supported
PostgreSQL 16	supported
PostgreSQL 15	supported
PostgreSQL 14	supported
PostgreSQL 13	supported
PostgreSQL 12	deprecated
PostgreSQL 11	deprecated
PostgreSQL 10	deprecated
MySQL	
MySQL 8	supported
MariaDB	

Database	Status
MariaDB 11.x	supported
MariaDB 10.x	supported
Oracle	
Oracle Database 21c	supported
Oracle Database 19c	supported
Oracle Database 18c	supported
Microsoft SQL Server	
MSSQL Server 2022	supported
MSSQL Server 2019	supported
MSSQL Server 2017	supported
IBM DB2	
IBM DB2 11.5	supported

2.1.2. Java

Formcentric Analytics kann mit den folgenden Java-Versionen betrieben werden. Die Ausführung muss innerhalb eines Servlet-Containers in Version 5.0 oder neuer erfolgen (z.B. Apache Tomcat 10).

Java	Status
OpenJDK 21	supported
OpenJDK 17	supported

2.1.3. Solr

Formcentric Analytics setzt eine Suchmaschine voraus. Wenn Sie sich für Solr entscheiden, finden Sie weiter Informationen in Abschnitt 2.5.2, „Solr“.

Solr	Status
Apache Solr 9.0.x - 9.8.x	supported
Apache Solr 8.11.x	deprecated

2.1.4. Elasticsearch

Formcentric Analytics setzt eine Suchmaschine voraus. Wenn Sie sich für Elasticsearch entscheiden, finden Sie weitere Informationen in Abschnitt 2.5.1, „Elasticsearch“.

Elasticsearch	Status
Elasticsearch 8.17.x	supported

Elasticsearch	Status
Elasticsearch 8.16.x	supported
Elasticsearch 8.15.x	supported

2.1.5. RabbitMQ / AMQP

Mithilfe von RabbitMQ können Sie eigene Anwendungen mit Formcentric Analytics integrieren. Die Verwendung ist optional.

RabbitMQ	Status
RabbitMQ 3.13	supported
RabbitMQ 3.12	supported
RabbitMQ 3.10	deprecated
RabbitMQ 3.9	deprecated

2.2. Installation

Die Backend-Webapp wird in zwei Varianten ausgeliefert. Sie können das Backend mit einem WAR in einem Servlet-Container betreiben oder als eigenständiges Spring-Boot-JAR.

Wir empfehlen die Verwendung des Spring-Boot-JARs. Dies bietet gegenüber eines WAR Deployments verschiedene Vorteile. Dazu gehören unter anderem:

- sowohl die Anwendung als auch Logging lassen sich einfacher konfigurieren (siehe unten),
- einfache Updates auf neue Versionen, da nur das JAR ausgetauscht werden muss,
- in einem Docker-Container lässt sich das JAR einfacher betreiben,
- es muss kein kompatibler Servlet-Container gewählt werden. In dem JAR ist ein geeigneter Tomcat Servlet-Container enthalten.

Jede Variante wird etwas anders konfiguriert. Die Konfiguration wird im Folgenden beschrieben.

2.2.1. Allgemeines

Unabhängig von der Variante sind folgende Konfigurationen vorzunehmen:

Schema einrichten
Legen Sie in Ihrer Datenbank ein Schema an, das durch das Formcentric Analytics Backend verwendet werden soll. Stellen Sie sicher, dass das Schema und die Datenbank das UTF-8 Encoding unterstützen, um Datenverluste und Encoding-Fehler zu verhindern.

Sollten Sie eine IBM DB2 Datenbank benutzen, stellen Sie bitte zusätzlich sicher, dass Sie einen Tabellenbereich mit einer Seitengröße (Pagesize) von mindestens 16 KB verwenden.

Datenbank-Benutzer anlegen
Erstellen Sie für das Schema einen Benutzer, der sämtliche CRUD-Operationen ausführen darf. Alle Tabellen werden beim ersten Start der Anwendung automatisch erstellt und bei Updates migriert, daher benötigt der Benutzer die Berechtigung, Tabellen innerhalb des Schemas zu verwalten.

Details zu den benötigten Zugriffsrechten finden Sie im Kapitel Abschnitt 2.7, „Notwendige Zugriffsrechte für den Datenbank-Benutzer“

Suchmaschine
Das Backend benötigt eine *Suchmaschine*. Sie können sich zwischen Solr und Elasticsearch entscheiden. Lesen Sie im Kapitel Abschnitt 2.5.2, „Solr“, wie Sie Solr einrichten und im Kapitel Abschnitt 2.5.1, „Elasticsearch“, wie Sie Elasticsearch einrichten.

2.2.2. Spring Boot

Dieser Abschnitt ist nur relevant, wenn Sie wie empfohlen das Spring-Boot-JAR verwenden möchten.

Zur Veranschaulichung der folgenden Abschnitte eine Darstellung der Verzeichnisse und Dateien:

```
.
|-- formcentric-backend-webapp-boot-${project.version}.jar
|-- logback-spring.xml
|-- application.properties
\-- lib/
    \-- your-jdbc-driver-here.jar
```

JDBC-Treiber bereitstellen
Die Formcentric Analytics Backend-Webapp bringt keine eigenen Datenbanktreiber mit. Um einen JDBC4 Treiber auf den Klassenpfad zu bringen, legen Sie ihn in einem Ordner */lib* neben das ausführbare JAR ab.

Logging
Die Spring-Boot-Variante vom Backend benutzt Logback. Zur Konfiguration empfehlen wir Ihnen, eine Datei mit dem Namen *logback-spring.xml* neben das ausführbare JAR abzulegen und die Datei in den *application.properties* (siehe unten) zu referenzieren.

Sie finden die verwendete Standard-Konfiguration in dem JAR unter */BOOT-INF/classes/logback-spring.xml*. Wir empfehlen Ihnen, diese als Startpunkt für Ihre Konfiguration zu verwenden.

Eine allgemeine Beschreibung der XML Konfiguration für Logback finden Sie unter folgender Adresse: https://logback.qos.ch/manual/configuration.html.

Konfiguration
Um das Backend zu konfigurieren, empfehlen wir Ihnen eine *application.properties* im gleichen Verzeichnis wie das JAR abzulegen. Alternativ können Sie die Parameter über Umgebungsvariablen setzen. Weitere Konfigurationsdetails des Backends finden Sie in Abschnitt 2.4, „Konfiguration“.

Der Inhalt der *application.properties* kann beispielsweise so aussehen:

```
# active database and search profile:
spring.profiles.active=postgres,solr

########################################################################
## Database settings
########################################################################

schema.name=mwf_analytics
spring.datasource.username=mwf_analytics
spring.datasource.password=Monday123
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://analytics-database:5432/${schema.name}

### Search & Solr Settings
analytics.solr.url=http://analytics-solr:8983/solr/
analytics.solr.snippet.count=50
analytics.solr.collection=mwfanalytics

########################################################################
## You can configure logging the 'Spring Boot way' (shown below) but
## we recommend to go with the logback-spring.xml configuration.
########################################################################

### configuration with an external logback-config-file:
logging.config=file:logback-spring.xml

### logging the spring boot way:
#logging.level.root=INFO
#logging.level.com.formcentric.backend=DEBUG
```

Hier einige Beispiele, wenn Sie das Backend per Umgebungsvariablen konfigurieren, was bei Containern eine beliebte Variante ist:

```
MANAGEMENT_SERVER_PORT=9090
MANAGEMENT_ENDPOINTS_WEB_EXPOSURE_INCLUDE=prometheus,health
ANALYTICS_ACTUATOR_SECURITY_ENABLED=false
```

Starten der Applikation
Die Anwendung kann in der gezeigten Konfiguration mit *java -Dloader.path="lib" -jar formcentric-backend-webapp-boot-4.5.2.jar* gestartet werden.

2.2.3. Servlet-Container

Dieser Abschnitt ist nur relevant, wenn Sie das Backend in einem Servlet-Container betreiben möchten.

JDBC-Treiber bereitstellen
Die Formcentric Analytics Backend-Webapp bringt keine eigenen Datenbanktreiber mit. Konfigurieren Sie Ihren Servlet-Container so, dass er die Treiber zur Verfügung stellt oder kopieren Sie ihren JDBC 4 Treiber in das Verzeichnis *WEB-INF/lib*, um ihn dem Classpath hinzuzufügen.

Logging
Konfigurieren Sie das Logging in der Datei *WEB-INF/classes/logback-spring.xml*.

Eine allgemeine Beschreibung der XML Konfiguration für Logback finden Sie unter folgender Adresse: https://logback.qos.ch/manual/configuration.html.

Konfiguration
Um das Backend zu konfigurieren, können Sie die WAR-Datei entweder entpacken und die Konfiguration unter *WEB-INF/classes/application-backend.properties* vornehmen oder die Parameter über Umgebungsvariablen setzen. Weitere Konfigurationsdetails des Backends finden Sie in Abschnitt 2.4, „Konfiguration“.

2.3. Monitoring und Management

In diesem Abschnitt wird beschrieben, welche Möglichkeiten es für das Monitoring der Applikation gibt und wie Sie hierfür die Einstellungen vornehmen.

Wenn die Einstellung *analytics.actuator.security.enabled* (siehe auch Abschnitt 2.4, „Konfiguration“) aktiv ist, muss sich der Aufrufer authentifizieren, um die Management-Endpoints aufgerufen zu können. Wird das Backend als Spring-Boot-JAR (siehe Abschnitt 2.2.2, „Spring Boot“) installiert, empfehlen wir Ihnen die Einstellung auf *false* zu setzen und stattdessen den Port des Management-Servers mit *management.server.port* vom Server-Port zu trennen. Dadurch vereinfachen Sie die Konfiguration von Monitoring-Systemen erheblich.

Stellen Sie sicher, dass niemand Unbefugtes den mit *management.server.port* festgelegten Port erreichen kann. Nutzen Sie hierfür am besten einen vorgeschalteten Proxy, der den Traffic nur auf den gewünschten Port (*server.port*) weitergibt.

Analytics setzt Spring Boot Actuator ein. Actuator ist ein ausgereiftes Management-Framework, das viele Konfigurationsmöglichkeiten bietet. Es folgen die Default-Einstellungen von Analytics:

management.endpoints.web.exposure.include: Hier wird angegeben, welche Endpunkte über *Web* verfügbar gemacht werden. Es können verschiedene Endpunkte veröffentlicht werden (siehe dazu auch https://docs.spring.io/spring-boot/docs/3.2.x/reference/html/actuator.html#actuator.endpoints). Der Default, mit dem Analytics ausgeliefert wird (*info,health,metrics*), wird in der folgenden Tabelle beschrieben:

…/actuator/info	Hier werden allgemeine Informationen der Applikation abgerufen, wie zum Beispiel: `{ "build": { "name": "formcentric-backend", "version": "${project.version}", "timestamp": "2023-01-19T09:09:04+0000" } }`
…/actuator/health	Hier wird der Health-Status der Applikation abgerufen. Dieser Endpunkt wird üblicherweise von Monitoring-Applikationen verwendet. In der Regel lautet die Ausgabe: `{ "status": "UP" }` Sollen mehr Details angezeigt werden, können Sie das konfigurieren (siehe https://docs.spring.io/spring-boot/docs/3.2.x/reference/html/actuator.html#actuator.endpoints.health).
…/actuator/metrics	Hier können verschiedene Metriken der Applikation abgerufen werden (siehe auch Abschnitt 2.3.1, „Metrics“).

management.info.env.enabled: Die Default-Einstellung ist *true*. Die allgemeinen Informationen, die über *…/actuator/info* (siehe oben) abgerufen werden, werden während des Builds in die Properties geschrieben.

management.health.rabbit.enabled: Die Default-Einstellung ist *${analytics.rabbitmq.enabled:false}*. Der Health-Status der RabbitMQ wird nur ausgewertet, wenn die RabbitMQ *enabled* ist.

management.health.mail.enabled: Die Default-Einstellung ist *${mail.sending.enabled:false}*. Der Health-Status des Mail-Servers wird nur ausgewertet, wenn das Senden von E-Mails aktiviert ist.

2.3.1. Metrics

Spring-Boot-Actuator liefert (*…/actuator/metrics*) sehr detaillierte Metriken der Applikation (siehe https://docs.spring.io/spring-boot/docs/3.2.x/reference/html/actuator.html#actuator.metrics.endpoint).

Rufen Sie die API ohne zusätzliche Parameter auf, wird eine Liste der verfügbaren Metriken zurückgegeben. Wenn Sie eine bestimmte Metrik überwachen möchten, dann hängen Sie den entsprechenden Namen an die URL an. Bei folgendem Beispiel wird die Metrik des benutzten Speichers abgerufen:

```
http://<host>/actuator/metrics/jvm.memory.used
```

Das Abrufen von nur einzelnen Werten ist ebenfalls möglich, wie im folgenden Beispiel dargestellt:

```
http://<host>/actuator/metrics/jvm.memory.used?tag=area:nonheap
```

2.3.2. Prometheus

Prometheus gehört mit zu den beliebtesten Monitoring-Systemen und wird von Spring-Boot-Actuator unterstützt.

Die Prometheus (https://prometheus.io/) Metriken-API ist standardmäßig nicht aktiviert. Fügen Sie *management.endpoints.web.exposure.include* den Wert *prometheus* hinzu, um sie zu aktivieren. Mehr Details zur Konfiguration finden Sie unter folgender Adresse: https://docs.spring.io/spring-boot/docs/3.2.x/reference/html/actuator.html#actuator.metrics.export.prometheus

2.4. Konfiguration

In diesem Abschnitt werden alle Konfigurationsparameter von Formcentric Analytics erläutert.

Als Erstes müssen Sie bestimmen, mit welcher Datenbank Sie arbeiten und welche Suchmaschine Sie verwenden:

spring.profiles.active: Mit diesem Konfigurationsparameter legen Sie fest, welche Datenbank-Konfiguration Sie für das Backend verwenden. Die möglichen Werte sind: *postgres*, *mysql*, *mariadb*, *db2*, *mssql* oder *oracle*. Zudem legen sie fest welche Suchmaschine vom Backend verwendet wird. Die möglichen Werte sind: *solr* oder *elastic*. Wenn Sie beispielsweise eine Oracle-Datenbank mit Elasticsearch verwenden, tragen Sie hier "*oracle,elastic*" ein.

Wenn Sie die Anwendung innerhalb von FirstSpirit betreiben, können Sie die Konfiguration im Server Manager vornehmen.

2.4.1. Datenbank

Die folgenden Parameter werden für die Verbindung zur Datenbank benötigt.

spring.datasource.username: Geben Sie hier den Benutzernamen des oben angelegten Benutzers ein.

spring.datasource.password: Geben Sie hier das Passwort des oben angelegten Benutzers ein.

spring.datasource.driver-class-name: Bitte tragen Sie hier den Class-Name des von Ihnen verwendeten jdbc-Treibers ein.

spring.datasource.url: Hinterlegen Sie hier Ihre JDBC-URL, die für Ihre Datenbank benötigt wird.

2.4.2. Allgemeine Konfiguration

Dieser Abschnitt beinhaltet allgemeine Konfigurationsparameter für Formcentric Analytics.

analytics.security.oauth2.clientSecret: Passwort, das Sie selbst erstellen und hinterlegen müssen. Hierbei ist zu beachten, dass dasselbe Secret in der Reporting-Webapp hinterlegt werden muss.

analytics.security.oauth2.analyticsClientSecret: Passwort, dass Sie selbst erstellen und hinterlegen müssen. Hiermit wird dem CMS der Zugriff auf Formcentric Analytics ermöglicht (siehe Abschnitt 2.6, „Zugriff aus dem Content-Management-System“).

analytics.security.oauth2.apiTokenValidity: Optional: Anzahl der Tage, die ein selbstgenerierter API-Token gültig ist. Wird -1 für die Gültigkeitsdauer vergeben, sind API-Token unbegrenzt gültig. (Default: *365*)

Falls die OAuth2 Client Secrets *analytics.security.oauth2.clientSecret* und *analytics.security.oauth2.analyticsClientSecret* nicht konfiguriert wurden, werden diese beim Start der Anwendung automatisch generiert und auf INFO geloggt.

analytics.actuator.security.enabled: Optional: Hier kann angegeben werden, ob die Spring Boot Actuator Endpunkte durch OAuth2 gesichert sind oder ob zum Abfragen keine Authentifizierung erforderlich ist. (Default: *true*)

analytics.registration.enabled: Hier kann eingestellt werden, ob die Möglichkeit sich selbst einen Analytics Account auf der Login Seite zu registrieren gegeben sein soll. (Default: *false*)

export.delimiter: Hier können Sie festlegen, mit welchem Zeichen Aufzählungen im CSV- oder Excel-Export getrennt werden. (Default: *;*)

export.textQualifier: Geben Sie hier den Textqualifizierer an. Mit diesem Zeichen werden Einträge, die das Trennzeichen enthalten, als Text markiert. (Default: *"*)

automation.cron: Cron-Ausdruck, der steuert, wann die Automatisierungsaufträge von Formcentric Analytics durchgeführt werden sollen. Die Standardkonfiguration führt die Aufträge jede Nacht um 1 Uhr durch. Der Cron-Ausdruck wird in UTC interpretiert.

automation.enabled: Hier können Sie das Ausführen von Automatisierungsaufträgen aktivieren / deaktivieren. (Default: *true*)

analytics.revisions.enabled: Hier können Sie das Bearbeiten von Formulareinträgen aktivieren / deaktivieren. (Default: *true*)

2.4.3. Mailversand

Die folgenden Parameter konfigurieren den Mailversand von Formcentric Analytics. Da sich der Mailversand auf Funktionen rund um die Benutzerverwaltung (Akti-

vierungslink, Passwort vergessen) beschränkt, ist ein Mailserver bei Verwendung eines externen Verzeichnisdienstes nicht zwingend erforderlich und kann deaktiviert werden.

mail.sending.enabled: Hier können Sie das Versenden von E-Mails aus Formcentric Analytics aktivieren / deaktivieren. (Default: *true*)

mail.user: Benutzer, mit dem sich Formcentric Analytics am Mailserver authentifiziert.

mail.password: Passwort für den Benutzer des Mailservers.

mail.host: Host des Mailservers.

mail.port: Port des Mailservers.

mail.properties.*: Alle weiteren JavaMail-Properties können mithilfe des Prefix *mail.properties* verwendet werden.

mail.properties.transport.protocol: Das Transportprotokoll des Mailservers.

mail.properties.smtp.auth: Aktivieren / Deaktivieren, ob sich der User mit dem AUTH-Befehl anmelden soll.

mail.properties.smtp.starttls.enable: Hier können Sie konfigurieren, ob START-TLS verwendet werden soll.

mail.properties.smtp.socketFactory.class: Java-Klasse zum Aufbau von SMTP Sockets. Entfernen Sie diesen Parameter für Plaintext-Verbindungen.

mail.properties.debug: Aktivieren / Deaktivieren von zusätzlichen Debugging Informationen während des Mailversands.

mail.from.address: Hier können Sie konfigurieren von welcher E-Mail-Adresse Formcentric Analytics E-Mails verschickt.

mail.from.name: Über diese Property vergeben Sie den Namen, der als Absender für alle E-Mails von Formcentric Analytics angezeigt werden soll.

mail.reporting.url: Geben Sie hier die URL zur Reporting-Komponente von Formcentric Analytics ein.

mail.language: Geben Sie an in welcher Sprache die E-Mails verschickt werden sollen.

2.4.4. Anbindung LDAP und Active Directory (Optional)

Durch die unten aufgeführten Parameter wird ein Lightweight Directory Access Protocol (LDAP) oder Active Directory Server an Formcentric Analytics zur Authentifizierung von Nutzern angebunden. Wenn wir später nur LDAP schreiben und Active Directory nicht explizit ausschliessen, ist dieser stets auch gemeint.

Damit sich externe Benutzer anmelden können, muss der Benutzer in mindestens einer LDAP-Gruppe sein, die Formcentric Analytics bekannt ist. Administratoren haben daher in der Systemverwaltung die Möglichkeit, LDAP-Gruppen zu synchronisieren. Schlagen alle Loginversuche fehl, ist dies eine häufige Fehlerursache und

sollte wiederholt werden. In der Gruppenverwaltung können Sie einer LDAP-Gruppe Berechtigungen und Rollen zuweisen, zum Beispiel um alle Mitglieder einer Gruppe zu Administratoren zu machen.

Falls Sie LDAP über SSL (LDAPs) einsetzen achten Sie darauf, das LDAP-Zertifikat Ihrem TrustStore hinzuzufügen.

Konfigurationsparameter des Servers: Die Serverparameter haben den prefix *analytics.ldap*. Dies gilt für auch für den Active Directory Server.

spring.profiles.include: Erweitern Sie diesen Parameter um die Angabe eines Spring-Profils, wenn Sie ein externes Benutzerverzeichnis verwenden möchten (z.B. `spring.profiles.include=db2,activeDirectory`). Folgende Profile stehen Ihnen hierfür zur Verfügung: *ldap*, *activeDirectory*.

analytics.ldap.userDn: Nutzername für den Zugriff auf den LDAP Server

analytics.ldap.password: Passwort für den Zugriff auf den LDAP Server

analytics.ldap.url: Url des LDAP Servers.

analytics.ldap.baseDn: Die Basis-DN des Benuzerverzeichnisses, von wo aus alle Benutzer und Gruppen erreicht werden können (z.B. `dc=mydomain,dc=com`).

analytics.ldap.subdomainDNs[0..n]: (Optional) Weitere Subdomain-DNs des Benuzerverzeichnisses, von wo aus Benutzer und Gruppen erreicht werden können (z.B. `dc=subdomain1,dc=mydomain,dc=com`). Für jede weitere Subdomain muss ein neuer Eintrag mit der voll qualifizierten DN der Subdomain eingefügt werden (z.B. `analytics.ldap.subdomainDNs[1]=dc=subdomain2,dc=mydomain,dc=com`)

analytics.ldap.domainLabel: Lesbare Bezeichnung der Domain, die dem Anwender bei der Anmeldung vorgeschlagen wird

analytics.ldap.userSearchBase: Angabe eines Objekts im Verzeichnisbaums, unterhalb dessen die Benutzersuche durchgeführt werden soll (z.B. `ou=people` für LDAP oder `cn=users` für Active Directory).

analytics.ldap.userSearchFilter: Angabe eines LDAP-Suchfilters, mit dem unterhalb der angegebenen Suchbasis nach Benutzern gesucht wird (z.B. `(uid={0})` für LDAP oder `(samaccountname={0})` für Active Directory).

Der Platzhalter *{0}* wird bei der Ausführung der Suche durch den eingegebenen Nutzernamen ersetzt.

Eine allgemeine Beschreibung der Syntax von Suchfiltern finden Sie unter folgender Adresse: http://www.faqs.org/rfcs/rfc2254.html.

analytics.ldap.userDnPattern: Pattern, mit dessen Hilfe der Distinguished Name (DN) eines Benutzers erzeugt werden kann. Diesen Parameter müssen Sie nur angeben, wenn Sie als Verzeichnistyp *LDAP* ausgewählt haben (z.B. `uid={0},ou=people`).

analytics.ldap.mailAttribute: Name des Attributfelds des Benutzer-Objekts, in dem die E-Mail-Adresse zu finden ist. Die E-Mail-Adresse wird in der Analytics Datenbank gespeichert.

analytics.ldap.groupSearchBase: Angabe eines Objektes im Verzeichnisbaums, unterhalb dessen die Gruppensuche durchgeführt werden soll (z.B. *DC=company,DC=com* für LDAP oder *cn=users* für Active Directory).

analytics.ldap.groupSearchFilter: Angabe eines LDAP-Suchfilters, mit dem unterhalb der angegebenen Suchbasis nach Gruppen gesucht wird (z.B. *(objectclass=groupOfUniqueNames)* für LDAP oder *(objectclass=group)* für Active Directory).

analytics.ldap.groupsImportFilter: Alle Gruppen, für die bestimmte Berechtigungen innerhalb von Formcentric Analytics vergeben werden sollen, müssen zuvor in das interne Benutzerverzeichnis von Formcentric Analytics importiert werden. Im Feld *Gruppenimportfilter* haben Sie die Möglichkeit einen LDAP-Suchfilter anzugeben, mit dem die zu importierenden LDAP-Gruppen aus der Liste der verfügbaren Gruppen ausgewählt werden (z.B. *(cn=*AnalyticsUsers*)*). Wenn Sie in diesem Feld nichts angeben, werden alle Gruppen importiert, die durch den im Feld *Suchfilter für Gruppen* angegeben LDAP-Suchfilter ermittelt werden.

analytics.ldap.userGroupsSearchFilter: Mit Hilfe dieses LDAP-Suchfilters werden die Gruppen eines Benutzers ermittelt, in denen er Mitglied ist. Diesen Parameter müssen Sie nur angeben, wenn Sie als Verzeichnistyp *LDAP* ausgewählt haben (z.B. *(memberUid={0})*). Der Platzhalter *{0}* wird bei der Ausführung der Suche durch den eingegebenen Nutzernamen ersetzt.

analytics.ldap.removeExternalUsers.enabled: Hier können Sie das automatische Entfernen von LDAP/AD Benutzern aus Analytics aktivieren (*true*) oder deaktivieren (*false*).

analytics.ldap.removeExternalUsers.cron: Cron-Ausdruck, der steuert, wann die LDAP/AD Benutzer aus Analytics entfernt werden. Es werden nur Benutzer entfernt, die im LDAP/AD nicht mehr gefunden werden können.

analytics.ldap.adBindDomainFromBaseDn: Betrifft nur Active Directory: Hier können Sie festlegen ob die *Domain* des Benutzers beim Anmelden aus dem Parameter *analytics.ldap.baseDn* ermittelt wird (*true*) oder nicht (*false*). Wenn Sie den Parameter nicht setzen ist der Default *true*.

analytics.ldap.adBindDomain: Betrifft nur Active Directory: Hier können Sie eine *Domain* hinterlegen, die falls *analytics.ldap.adBindDomainFromBaseDn* auf *false* steht immer eingesetzt wird. Wenn Sie diesen Parameter nicht setzen muss der Benutzer seine Domain beim Anmelden selbst mit angeben.

2.4.5. Anbindung SSO / OpenId-Connect (Optional)

In diesem Abschnitt wird erläutert, wie OIDC (OpenId-Connect) konfiguriert wird. Durch die Nutzung von OIDC können Sie Single-Sign-On (SSO) realisieren.

Beachten Sie die folgenden Auswirkungen bei der Nutzung von OIDC für Analytics:

- Sie können OIDC nicht gleichzeitig mit LDAP und Active Directory verwenden.

- Durch die Nutzung von OIDC verwaltet Analytics selbst keine Benutzer mehr. Alle Benutzerinformationen stammen aus dem JWT (JSON-Web-Token).
- Analytics stellt keine eigenen OAuth2 Tokens mehr aus. Ihre Tokens kommen ausschließlich von Ihrem OIDC-Server. Die API-Zugangsverwaltung entfällt dadurch. Jeder Zugriff mit einem gültigen JWT erhält entsprechende Rechte auf die Analytics-Backend-Schnittstelle.
- Analytics erzeugt Gruppen automatisch, wenn sie im JWT enthalten sind und als Analytics-Gruppe erkannt werden. Wenn Sie Gruppen löschen, werden diese automatisch wieder angelegt, sobald ein Benutzer dieser Gruppe erkannt wird.
- Sie können Formularrechte nur noch Gruppen zuordnen, nicht mehr einzelnen Benutzern.
- In der Reporting-Oberfläche werden für OIDC nicht relevante Menüpunkte ausgeblendet (z.B. die Benutzer-Verwaltung).

Die meisten Einstellungen müssen Sie für das Backend vornehmen. Die Einstellungen für das Reporting finden Sie unter: Abschnitt 3.4.2, „Anbindung SSO / OpenId-Connect (Optional)“.

Spring

spring.profiles.include: Zur Aktivierung von OIDC müssen Sie hier das Spring-Profil *oidc* hinzufügen (z.B. `spring.profiles.include=db2,solr,oidc`).

spring.security.oauth2.resourceserver.jwt.issuer-uri: Tragen Sie hier die URI Ihres Authorisierungs-Servers ein, die in Ihren JWTs unter *iss* steht. Ihr Authorisation-Server muss einen Provider Configuration Endpunkt (siehe auch https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig) bereit stellen (auch "Authorization Server Metadata" Endpunkt genannt: https://datatracker.ietf.org/doc/html/rfc8414#section-3). Sollte dies nicht der Fall sein, wenden Sie sich bitte an unseren Support. Wir helfen Ihnen bei der Konfiguration.

Bei der Untersuchung des JWTs schaut Analytics nur in die *Claims* (die Payload), nicht in den Header.

Alle Einstellungen, die einen Pfad beschreiben sind ein JSON-Path Ausdruck. Alle Pfad-Einstellungen sind JSON-Path-Ausdrücke. Sie können mehrere Ausdrücke durch Kommas trennen. Das standardmäßige führende *$* können Sie in der Analytics-Konfiguration weglassen. Analytics erwartet unter diesen Pfaden entweder String-Arrays oder einzelne Strings; andere Parametertypen werden ignoriert.

Um die Konfiguration zu vereinfachen, empfehlen wir Ihnen, einen JWT von Ihrem Server zu decodieren (z.B. mit https://jwt.io/) und die Payload in ein Online-Tool zu geben, mit dem die JSON-Pfade getestet werden können (z.B. https://jsonpath.com/).

Gruppen

Es folgen die Einstellungen, mit denen Analytics feststellt, in welchen Gruppen der Besitzer des Tokens ist:

analytics.security.oidc.groups-claim-paths: Definieren Sie hier einen oder mehrere JSON-Path-Ausdrücke, mit denen Analytics Gruppen im JWT findet.

analytics.security.oidc.group-filter-prefixes: Mit dieser optionalen Einstellung können Sie gefundene Gruppen nach Präfix filtern. Wenn Sie beispielsweise Analytics als Präfix festlegen, werden nur Gruppen berücksichtigt, deren Name mit *analytics_* beginnt.

analytics.security.oidc.admin-group-name: Mit dieser Einstellung legen Sie den Namen der Admin-Gruppe fest. Analytics benötigt diese Information, um zu bestimmen, welche Benutzer Administratoren sind. Dies geschieht über die Rolle *admin*, die automatisch der konfigurierten Gruppe zugewiesen wird. Alle Benutzer in dieser Gruppe erhalten die Administrator-Rolle. Nachdem Analytics die Gruppen identifiziert und gefiltert hat, wird der Präfix vor dem Gruppennamen entfernt. Wenn der verbleibende Name mit dem hier festgelegten Namen übereinstimmt, wird der Benutzer der Admin-Gruppe zugeordnet. Ein Beispiel:

```
# 1. Gruppen, die ueber `...oidc.groups-claim-paths` gefundene wurden:
#    ['ana_admins', 'ana_foobar']
# 2. Gruppen, nachdem filtern (`...oidc.group-filter-prefixes=ana_`):
#    ['admins', 'foobar']
analytics.security.oidc.admin-group-name=admins
# Analytics legt automatisch eine Gruppe namens `admins` an,
# der die Rolle `admin` zugewiesen wird.
```

analytics.security.oidc.user-group-name: Hier legen Sie den Namen der User-Gruppe fest. Diese Gruppe erhält automatisch die Rolle *user*, und alle Benutzer in dieser Gruppe bekommen ebenfalls die User-Rolle. Ohne diese Rolle kann ein Benutzer in Analytics keine Aktionen durchführen (außer er ist Administrator). Nachdem Analytics die Gruppen ermittelt und gefiltert hat, wird der Präfix vor dem Gruppennamen entfernt. Wenn der verbleibende Name mit dem hier festgelegten Namen übereinstimmt, wird der Benutzer der User-Gruppe zugeordnet. Diese Funktionalität entspricht im Wesentlichen der von *analytics.security.oidc.admin-group-name*.

Rollen

Analytics arbeitet mit Rollen. Diese Rollen werden einem Benutzer zugeordnet, was ähnlich zur Zuordnung der Gruppen passiert. Analytics kennt folgende Rollen:

Rolle ***admin*****:** Benutzer mit dieser Rolle sind Administratoren und haben umfassende Rechte.

Rolle ***user*****:** Benutzer mit dieser Rolle sind normale Benutzer mit eingeschränkten Rechten.

Rolle *api_consumer***:** Diese Rolle ist für Maschinen gedacht. Ein Beispiel für einen solchen Consumer ist Formcentric. Es ist jedoch auch möglich, dass Sie ein eigenes Programm entwickeln, das mit dieser Rolle auf die API zugreift.

Wir empfehlen, Benutzern nicht direkt die Rollen 'admin' oder 'user' zuzuweisen, sondern dies über Gruppen zu konfigurieren. Der Ansatz über Gruppen ist deutlich flexibler, da Sie Rechte auf Gruppenebene zuweisen können, was bedeutet, dass alle Mitglieder dieser Gruppen die gleichen Rechte erhalten. Das Anlegen von Rollen-Mappings ist optional. Sie müssen jedoch ein Mapping für Formcentric erstellen, da Formcentric ansonsten keine Daten in Analytics schreiben kann.

Es folgen die Einstellungen, mit denen Analytics feststellt, mit welchen Rollen der Besitzer des Tokens auszustatten ist:

analytics.security.oidc.role-claim-paths: Wie beim Suchen nach Gruppen (siehe *analytics.security.oidc.groups-claim-paths*) schaut Analytics hier mit einem oder mehreren JSON-Path-Ausdrücken im JWT nach Rollen.

analytics.security.oidc.role-filter-prefixes: Wie bei den Gruppen können Sie die Rollen mit einem Präfix filtern. So können Sie z.B. mit dem Präfix Analytics so konfigurieren, dass nur Rollen berücksichtigt werden, deren Name mit *analytics_* anfängt. Diese Konfiguration ist optional.

analytics.security.oidc.role-mappings.<role_name>: Hier wird der Rollenname auf die Analytics-Rolle gemappt. Analytics ermittelt zunächst die Rollen aus dem JWT und filtert dann die gefundenen Namen mit dem Präfix. Der Präfix wird anschließend vom Namen abgezogen. Ein Beispiel:

```
# 1. Rollen, die ueber `...oidc.role-claim-paths` gefunden wurden:
#    ['ana_role_administrator', 'ana_role_consumer', 'somthing_different']
# 2. Rollen nach dem Filtern (`...oidc.role-mappings.admin=ana_role`):
#    ['administrator', 'consumer']
# 3. Zuordnung der Rollen in Analytics:
#    analytics.security.oidc.role-mappings.<role_name>
analytics.security.oidc.role-mappings.administrator=admin
analytics.security.oidc.role-mappings.consumer=api_consumer
```

In dem Beispiel werden zwei Rollen-Mappings vorgenommen:

1. Die Rolle *administrator* aus dem JWT wird auf die Rolle *admin* von Analytics gemappt.

2. Die Rolle *consumer* aus dem JWT wird auf die Rolle *api_consumer* von Analytics gemappt.

Die Groß- und Kleinschreibung des Analytics-Rollennamens wird ignoriert. Es ist auch möglich, ein führendes *role_* anzugeben. Das bedeutet, dass folgende Schreibweisen äquivalent sind: *role_api_consumer*, *api_consumer*, *Api_Consumer*, *API_CONSUMER*, *ROLE_API_CONSUMER*, usw.

Das Analytics-Backend liefert dem Reporting die Benutzerdaten. Damit im Reporting der Benutzername und die E-Mail-Adresse korrekt angezeigt werden, gibt es

die folgenden optionalen Einstellungen. Wenn hier keine Konfiguration vorgenommen wird, erscheinen die entsprechenden Daten nicht im Reporting.

analytics.security.oidc.preferred-user-name-path: Mit diesem JSON-Path wird in dem JWT nach dem Benutzernamen gesucht. Wird ein String-Array gefunden, zeigt das Reporting den ersten String als Benutzernamen an.

analytics.security.oidc.email-path: Mit diesem JSON-Path wird in dem JWT nach der E-Mail-Adresse des Benutzers gesucht. Wird ein String-Array gefunden, zeigt das Reporting den ersten String als E-Mail-Adresse an.

2.4.6. RabbitMQ / AMQP (Optional)

Mithilfe von RabbitMQ können Sie eigene Anwendungen mit Formcentric Analytics integrieren.

analytics.rabbitmq.enabled: Hier können Sie die RabbitMQ Integration aktivieren (*true*) oder deaktivieren (*false*).

analytics.rabbitmq.topic.enabled: Hier können Sie den Topic Exchange Type aktivieren (*true*) oder deaktivieren (*false*).

analytics.rabbitmq.headers.enabled: Hier können Sie den Headers Exchange Type aktivieren (*true*) oder deaktivieren (*false*).

spring.rabbitmq.host: Hier können Sie den RabbitMQ Host konfigurieren. Des Weiteren stehen alle von Spring RabbitMQ unterstützten Konfigurationsparameter zur Verfügung.

spring.rabbitmq.port: Hier können Sie den RabbitMQ Port konfigurieren. Des Weiteren stehen alle von Spring RabbitMQ unterstützten Konfigurationsparameter zur Verfügung.

spring.rabbitmq.username: Hier können Sie den RabbitMQ Benutzernamen konfigurieren. Des Weiteren stehen alle von Spring RabbitMQ unterstützten Konfigurationsparameter zur Verfügung.

spring.rabbitmq.password: Hier können Sie das RabbitMQ Passwort konfigurieren. Des Weiteren stehen alle von Spring RabbitMQ unterstützten Konfigurationsparameter zur Verfügung.

2.4.7. Request-ID

Formcentric Analytis kann eine Request-ID aus den Http-Request-Headern entnehmen und ins Log ausgeben. Dies ermöglicht die Verfolgung eines Requests über mehrere Logeinträge hinweg. Formcentric Analytics setzt dazu eine Variable *REQ-ID* in den Mapped Diagnostic Context (MDC) des verwendeten Logging-Frameworks (siehe Abschnitt 2.2.2, „Spring Boot“ bzw. Abschnitt 2.2.3, „Servlet-Container“).

Mit den folgenden Parametern können Sie diese Funktionalität konfigurieren.

analytics.http.requestId.enabled: Dieser Parameter ermöglicht es Ihnen, diese Funktion zu aktivieren (*true*) oder zu deaktivieren (*false*). Standardmäßig ist diese Funktionalität aktiviert.

analytics.http.requestId.headerName: Mit diesem Parameter können Sie den Namen des Http-Headers konfigurieren, aus dem Formcentric Analytics die Request-ID entnimmt. Standardmäßig ist der Wert *x-request-id*.

analytics.http.requestId.internalRequestIdPrefix: Wenn der Request keine Request-ID enthält, vergibt Formcentric Analytics automatisch eine Request-ID. Mit diesem Parameter können Sie den Präfix dieser automatisch generierten ID festlegen, um die ID im Log besser unterscheiden zu können.

analytics.http.requestId.externalRequestIdPrefix: Formcentric Analytics kann einen Präfix vor die externe Request-ID (die ID aus dem Request-Header) setzen, um im Log deutlich zu machen, dass diese ID nicht von Formcentric Analytics vergeben wurde.

2.4.8. Verschlüsselung von Konfigurationsparametern

In der Grundkonfiguration werden sensitive Daten im Klartext in Konfigurationsdateien gespeichert. Bei einem eventuellen Einbruch können so gültige Zugangsdaten entwendet werden. Aus diesem Grund haben Sie die Möglichkeit Passwörter auch verschlüsselt abzulegen. Die Passwörter werden in diesem Fall erst beim Start der Anwendung mit dem hinterlegten Verschlüsselungspasswort entschlüsselt. Das zur Verschlüsselung verwendete Passwort müssen Sie vor dem Start der Anwendung in einer Umgebungsvariable speichern. Standardmäßig wird hiefür die Umgebungsvariable *MWF_ENCRYPTION_PASSWORD* verwendet.

Um Werte zu ver- und entschlüsseln gibt es ein CommandLine Tool, das mit dem Befehl

```
mvn dependency:copy -Dartifact=com.formcentric:encryption-cli:2.0:jar
      -DoutputDirectory=.
```

bezogen werden kann. Bei einem Aufruf wird es entweder einen Wert ver- oder entschlüsseln.

```
java -jar encryption-cli-1.0.jar -p 'encryptionPassword'
      -e 'toEncrypt'
```

Falls es *ohne* Parameter gestartet wird, wird es nach dem Passwort und den zu ver- bzw. entschlüsslenden Werten fragen. Bitte beachten Sie dass die Parameter in einfachen Anführungszeichen angegeben werden müssen. Es kann mit den folgenden Parametern gestartet werden:

Parameter	**Beschreibung**
-p oder --encryption-Password	Um das Passwort anzugeben das zur Ver- oder Entschlüsselung verwendet wird.
-d oder --decrypt	Um den darauf folgenden Wert zu entschlüsseln
-e oder --encrypt	Um den darauf folgenden Wert zu verschlüsseln
-? oder --help	Um die Hilfe zu dem Tool anzuzeigen.

2.5. Suchmaschine

Für alle Suchvorgänge wird eine Suchmaschine benötigt. Sie können zwischen den folgenden beiden Suchmaschinen wählen.

2.5.1. Elasticsearch

Im Folgenden erfahren Sie, wie Sie Elasticsearch betreiben können.

Installation Elasticsearch

Elasticsearch wird als eigenständige Anwendung betrieben. Eine Anleitung zur Installation und Konfiguration von Elasticsearch finden Sie auf der Produktseite https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.

Analytics erstellt den Index in Elasticsearch automatisch, sollte dieser noch nicht vorhanden sein. Die meisten Index-Konfigurationen können auch nach der Erstellung noch angepasst werden.

Alternativ zur automatischen Index-Erstellung können Sie den Index auch vorab manuell anlegen. In diesem Fall erkennt Analytics beim Start den bestehenden Index und verwendet diesen für alle Operationen.

Orientieren Sie sich an dem folgenden Beispiel, um den Index selbst anzulegen:

```
PUT http://localhost:9200/analytics
Authorization: Basic elastic elastic
Content-Type: application/json

{
    "aliases": {},
    "mappings": {
        "dynamic": true,
        "dynamic_templates": [
            {
                "double-mapping-always": {
                    "match_mapping_type": [ "double" ],
                    "mapping": { "type": "double" }
                }
            }
        ],
        "properties": {
            "formId": { "type": "keyword" },
            "dtFieldValues": { "type": "keyword", "store": true },
            "attachments": { "type": "binary" },
            "numbers": {
                "type": "nested",
                "properties": {
                    "_class": { "type": "keyword", "doc_values": false,
                        "index": false },
                    "value": { "type": "double" },
                    "key": { "type": "keyword" }
                }
            },
            "dates": {
```

```
                "type": "nested",
                "properties": {
                    "_class": { "type": "keyword","doc_values": false,
                        "index": false },
                    "value": { "type": "long" },
                    "key": { "type": "keyword" }
                }
            },
            "fieldValues": { "type": "text", "store": true },
            "recordId": { "type": "keyword", "index": true },
            "versionId": { "type": "keyword" },
            "statusId": { "type": "keyword" },
            "form": {
                "type": "nested",
                "properties": {
                    "_class": { "type": "keyword", "doc_values": false,
                        "index": false },
                    "value": { "type": "wildcard" },
                    "key": { "type": "keyword" }
                }
            },
            "meta": {
                "type": "nested",
                "properties": {
                    "_class": { "type": "keyword", "doc_values": false,
                        "index": false },
                    "value": { "type": "wildcard" },
                    "key": { "type": "keyword" }
                }
            },
            "_class": { "type": "keyword", "doc_values": false,
                "index": false },
            "inputMap": { "type": "binary" },
            "tenant": { "type": "keyword" }
        }
    },
    "settings": {
        "index": {
            "number_of_shards": "1",
            "number_of_replicas": "0",
            "refresh_interval": "1s"
        }
    }
}
```

Analytics geht davon aus, dass ein bestehender Index mit dem richtigen Namen korrekt konfiguriert ist. Das oben gezeigte Mapping muss daher exakt wie dargestellt übernommen werden.

Nachdem die Elasticsearch bereitsteht, müssen Sie üblicherweise Folgendes konfigurieren:

analytics.elastic.index: Hier können Sie den Namen des Indexes in Elasticsearch bestimmen. Der Standard-Name ist *analytics*.

spring.elasticsearch.uris: Eine durch Kommata getrennte Liste der zu verwendenden Elasticsearch-Instanzen. Der Standard ist *localhost:9200*

spring.elasticsearch.username: Benutzername für die Authentifizierung mit Elasticsearch. Der Standard-Benutzername ist *elastic*.

spring.elasticsearch.password: Passwort für die Authentifizierung mit Elasticsearch. Das Standard-Passwort ist *elastic*.

SSL

Dieser Abschnitt ist nur relevant, wenn Sie SSL einrichten möchten.

Analytics benutzt die Spring-Boot-Integration für Elasticsearch. SSL wird entsprechend per Spring-Boot konfiguriert (siehe https://docs.spring.io/spring-boot/reference/features/ssl.html).

Aufgrund der vielfältigen Konfigurationsmöglichkeiten zeigen wir Ihnen exemplarisch die Konfiguration eines *PKCS12* Keystores:

```
spring.ssl.bundle.jks.mybundle.key.alias=application
spring.ssl.bundle.jks.mybundle.keystore.location=/path/to/your/keystore.p12
spring.ssl.bundle.jks.mybundle.keystore.password=secret
spring.ssl.bundle.jks.mybundle.keystore.type=PKCS12
```

Den Namen *mybundle* können Sie frei wählen. Dieser Name muss anschließend in der Elasticsearch-Integration konfiguriert werden, zum Beispiel:

```
spring.elasticsearch.restclient.ssl.bundle=mybundle
```

2.5.2. Solr

Im Folgenden erfahren Sie, wie Sie Solr betreiben können.

Installation Solr

Die Solr-Suchmaschine läuft als eigenständige Anwendung. Eine Anleitung zur Installation und Konfiguration finden Sie auf der offiziellen Produktseite http://lucene.apache.org/solr/.

Um die externe Solr-Instanz mit Formcentric Analytics verwenden zu könnnen, müssen Sie zunächst einen neuen Core mit dem Namen *mwfanalytics* anlegen. Sie können auch den Namen des Cores mit *analytics.solr.collection* festlegen.

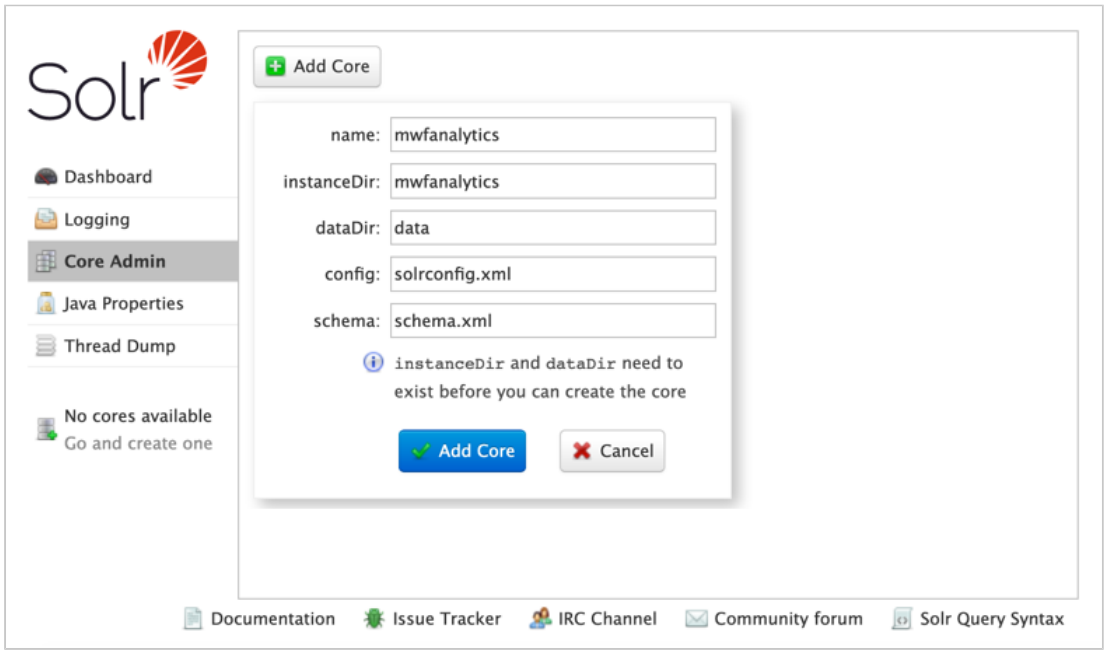

Abbildung 2.1. Core Administration in der Solr-Administrationsoberfläche

Die erforderlichen Konfigurationsdateien *solrconfig.xml* und *schema.xml* können Sie von unserem Maven-Server herunterladen:

```
mvn dependency:copy \
    -Dartifact=com.formcentric.analytics:formcentric-backend-solr-config:4.5.2:tar \
    -DoutputDirectory=.
```

Tragen Sie anschließend die URL des Solr-Servers in die Datei *WEB-INF/classes/application-backend.properties* im Parameter *analytics.solr.url* ein.

Beispiel: `analytics.solr.url=http://localhost:8983/solr/`

Allgemeine Einstellungen

analytics.solr.user: Hier können Sie den Benutzernamen für die Verbindung zur Solr angeben.

analytics.solr.pass: Hier können Sie das Passwort für die Verbindung zur Solr angeben.

analytics.solr.snippet.count: Maximale Anzahl von Trefferstellen, die für ein Suchergebnis ermittelt werden.

SSL

Für die Einrichtung von SSL können Sie folgende Parameter verwenden:

analytics.solr.ssl.keyStoreType: Definiert den Typ des Keystores. Mögliche Werte sind (siehe auch https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#KeyStore): *jceks*, *jks*, *dks*, *pkcs11* oder *pkcs12* (Default)

analytics.solr.ssl.keyStore: Gibt die zu verwendende Keystore-Datei an. Wir empfehlen, einen absoluten Pfad zu verwenden.

analytics.solr.ssl.keyStorePass: Legt das Passwort für den Keystore fest. Wenn der Keystore kein Passwort besitzt, ist diese Einstellung nicht erforderlich.

2.5.3. Feeder

Der Feeder synchronisiert die Datenbankinhalte mit der Suchmaschine. Die folgenden Konfigurationen können Sie für den Feeder vornehmen:

feeder.enabled: Aktiviert oder deaktiviert den Feeder. Standardmäßig ist er aktiviert. Diese Einstellung eignet sich, wenn bei mehreren gleichzeitig betriebenen Backends nur eines die Suchmaschine aktualisieren soll.

feeder.id: Hier können Sie eine eindeutige ID für den Feeder festlegen, die als beliebige Zeichenkette definiert werden kann. Bei mehreren Backend-Anwendungen empfehlen wir, unterschiedliche Feeder-IDs zu verwenden, um Konflikte zu vermeiden.

2.6. Zugriff aus dem Content-Management-System

Für den Zugriff auf die Backend-Webapp wird ein *Access-Token* benötigt. Die Generierung des Tokens erfolgt mithilfe des *analytics.security.oauth2.analyticsClientSecret* (siehe Abschnitt 2.2, „Installation“).

Es gibt drei verschiedene Möglichkeiten, den Token zu generieren:

1. Backend-Webapp

   Für die erste Variante wird ein konfiguriertes und laufendes Backend System benötigt. Aus der Antwort des Servers wird lediglich der Wert des *access_token* benötigt.

   ```
   curl --user webforms-webapp-client:${WEBFORMS_CLIENT_SECRET}
         ${BACKEND_HOST}/oauth/token -d grant_type=client_credentials
   ```

   Ergebnis:

   ```
   {"access_token":"uFDXjCR+pWL+8iQIigJr9iFLcQm04G7NILrrY+bWcHg=", ...}
   ```

2. Reporting-Webapp

   Die zweite Variante benötigt die Backend-Webapp sowie die Reporting-Webapp. Administratoren können den webforms-webapp-client in der Systemübersicht der Reporting-Webapp analog zu selbst angelegten API Zugängen konfigurieren und Token anfordern oder invalidieren. Eine detaillierte Anleitung finden Sie im Benutzerhandbuch der Reporting-Webapp.

3. Automatische Generierung im CMS

Anstelle eines zur Laufzeit der Backend-Webapp generierten Tokens können Sie der Formcentric CMS-Erweiterung auch das *Client Secret* übergeben. In diesem Falle fordert der *BackendApiClient* beim ersten Zugriff auf Formcentric Analytics automatisch ein Token an.

2.7. Notwendige Zugriffsrechte für den Datenbank-Benutzer

Der Datenbank-Benutzer benötigt die Berechtigung, Tabellen innerhalb des Schemas verwalten zu können, da beim ersten Start der Anwendung alle Tabellen automatisch erstellt und bei Updates migriert werden.

Neben den CRUD Operationen (*INSERT*, *SELECT*, *UPDATE*, *DELETE*) werden Berechtigungen für DDL-Operationen benötigt. Folgende Operationen muss der Datenbank-Benutzer ausführen können: *CREATE TABLE*, *ALTER TABLE*, *DROP TABLE*, *CREATE INDEX*, *DROP INDEX*, *CREATE PROCEDURE*, *DROP PROCEDURE*. Außerdem muss er berechtigt sein, Datenbank-Prozeduren ausführen zu können.

Einige Datenbanken arbeiten mit Sequenzen (DB2, Oracle, Postgres, MS-SQL). Für diese Datenbanken muss der Datenbank-Benutzer berechtigt sein, Sequenzen anzulegen und zu löschen (*CREATE SEQUENCE*, *DROP SEQUENCE*).

2.7.1. DB2

Während das Schema der Datenbank migriert wird, müssen in DB2 Tabellen neu organisiert werden. Hierfür muss der Datenbank-Benutzer folgende Operation durchführen können: *call SYSPROC.ADMIN_CMD('REORG TABLE <table>');*

2.7.2. Oracle

Bei Oracle werden bei einigen Operationen Objekte in einen Papierkorb gelegt. Dieser Papierkorb wird mit folgendem Kommando geleert: *purge recyclebin;*

Während der Migration des Schemas muss PL/SQL Code ausgeführt werden. Der Datenbank-Benutzer muss Code wie im Folgenden dargestellt ausführen können:

```
-- alter table MWF_FORM_SEARCH_NUMBER drop column ID if exist
DECLARE
    cnt integer;
BEGIN
    SELECT COUNT(*)
    INTO cnt
    FROM ALL_TAB_COLUMNS
    WHERE table_name = 'MWF_FORM_SEARCH_NUMBER'
      AND column_name = 'ID';

    IF (cnt = 1) THEN
        EXECUTE IMMEDIATE
            'alter table MWF_FORM_SEARCH_NUMBER drop column ID';
```

```
    END IF;
END;
```

2.7.3. MS-SQL

Während der Migration des Schemas muss Transact-SQL Code ausgeführt werden. Der Datenbank-Benutzer muss Code wie im Folgenden dargestellt ausführen können:

```
declare @var1 AS nvarchar(256);
select @var1 = TC.CONSTRAINT_NAME
from INFORMATION_SCHEMA.TABLE_CONSTRAINTS TC
inner join INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE CC on
  TC.CONSTRAINT_NAME = CC.CONSTRAINT_NAME
where CC.COLUMN_NAME = 'attachment_id'
  and CC.TABLE_NAME = 'mwf_form_attachments'
  and TC.CONSTRAINT_TYPE = 'PRIMARY KEY'
execute ('ALTER TABLE mwf_form_attachments DROP CONSTRAINT ' + @var1);
alter table mwf_form_attachments
  add constraint form_attachments_pk primary key (attachment_id, record_id);
```

Oder:

```
declare @sql nvarchar(200)
declare @constName nvarchar(100)
set @constName = (
  select top 1 TC.CONSTRAINT_NAME
  from INFORMATION_SCHEMA.TABLE_CONSTRAINTS TC
  inner join INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE CC on
    TC.CONSTRAINT_NAME = CC.CONSTRAINT_NAME
  where TC.CONSTRAINT_TYPE = 'UNIQUE'
  and COLUMN_NAME='email'
  and TC.TABLE_NAME='mwf_user')
print @constName
set @sql = 'ALTER TABLE mwf_user DROP CONSTRAINT '+ @constName
Exec sp_executesql @sql;
```

2.7.4. Postgres

Während der Migration des Schemas muss PL/SQL Code ausgeführt werden. Der Datenbank-Benutzer muss Code wie im Folgenden dargestellt ausführen können:

```
DO $$DECLARE r record;
BEGIN
  FOR r IN
    SELECT
      tc.constraint_name, tc.table_name
    FROM
      information_schema.table_constraints AS tc
      JOIN information_schema.key_column_usage AS kcu
        ON tc.constraint_name = kcu.constraint_name
      JOIN information_schema.constraint_column_usage AS ccu
        ON ccu.constraint_name = tc.constraint_name
    WHERE constraint_type = 'UNIQUE'
      AND tc.table_name='mwf_user'
```

```
      AND (ccu.column_name='user_name' OR ccu.column_name='email')
  LOOP
    EXECUTE 'ALTER TABLE ' || quote_ident(r.table_name) ||
      ' DROP CONSTRAINT ' || quote_ident(r.constraint_name) ||
      ';';
  END LOOP;
END$$;
```

3. Reporting-Webapp

Im diesem Kapitel erhalten Sie umfassende Informationen zu den erforderlichen System- und Browservoraussetzungen sowie eine detaillierte Anleitung zur Installation und Konfiguration der Formcentric Analytics Reporting-Webapp.

3.1. Systemvoraussetzungen

- Formcentric Analytics Backend 4.5.2

3.1.1. Java

Formcentric Analytics kann mit den folgenden Java-Versionen betrieben werden. Die Ausführung muss innerhalb eines Servlet-Containers in Version 5.0 oder neuer erfolgen (z.B. Apache Tomcat 10) oder Sie führen das Spring-Boot JAR aus (empfohlen).

Java	Status
OpenJDK 21	supported
OpenJDK 17	supported

3.2. Browservoraussetzungen

- Google Chrome (aktuellste Version)
- Mozilla Firefox (aktuellste Version oder ESR)
- Microsoft Edge (aktuellste Version)

3.3. Installation

Die Reporting-Webapp wird in zwei Varianten ausgeliefert. Sie können das Reporting mit einem WAR in einem Servlet-Container betreiben oder als eigenständiges Spring-Boot-JAR.

Wir empfehlen die Verwendung des Spring-Boot-JARs. Dies bietet gegenüber eines WAR Deployments verschiedene Vorteile. Dazu gehören unter anderem:

- sowohl die Anwendung als auch Logging lassen sich einfacher konfigurieren (siehe unten),
- einfache Updates auf neue Versionen, da nur das JAR ausgetauscht werden muss,
- in einem Docker-Container lässt sich das JAR einfacher betreiben,

- es muss kein kompatibler Servlet-Container gewählt werden. In dem JAR ist ein geeigneter Tomcat Servlet-Container enthalten.

Sie können überprüfen, ob die Installation erfolgreich war, indem Sie die Reporting-Webapp aufrufen und sich anmelden. Nach der initialen Installation lautet der Benutzername *admin* und das Passwort *admin*.

3.3.1. Spring-Boot

Dieser Abschnitt ist nur relevant, wenn Sie wie empfohlen das Spring-Boot-JAR verwenden möchten.

Logging
Die Spring-Boot-Variante vom Reporting benutzt Logback. Zur Konfiguration empfehlen wir Ihnen, eine Datei mit dem Namen *logback-spring.xml* neben das ausführbare JAR abzulegen und die Datei in den *application.properties* (siehe unten) zu referenzieren.

Sie finden die verwendete Standard-Konfiguration in dem JAR unter */BOOT-INF/classes/logback-spring.xml*. Wir empfehlen Ihnen, diese als Startpunkt für Ihre Konfiguration zu verwenden.

Eine allgemeine Beschreibung der XML Konfiguration für Logback finden Sie unter folgender Adresse: https://logback.qos.ch/manual/configuration.html.

Konfiguration
Um das Reporting zu konfigurieren, empfehlen wir Ihnen eine *application.properties* im gleichen Verzeichnis wie das JAR abzulegen. Alternativ können Sie die Parameter über Umgebungsvariablen setzen. Weitere Konfigurationsdetails des Backends finden Sie in Abschnitt 2.4, „Konfiguration“.

Der Inhalt der *application.properties* kann beispielsweise so aussehen:

```
### Backend Connection Settings
analytics.backend.url=http://analytics-backend:8080/

#########################################################################
## You can configure logging the 'Spring Boot way' (shown below) but
## we recommend to go with the logback-spring.xml configuration.
#########################################################################

logging.config=file:logback-spring.xml

### logging the spring boot way:
#logging.level.root=INFO
#logging.level.com.formcentric=DEBUG
```

Hier einige Beispiele, wenn Sie das Backend per Umgebungsvariablen konfigurieren, was bei Containern eine beliebte Variante ist:

```
MANAGEMENT_SERVER_PORT=9090
```

```
MANAGEMENT_ENDPOINTS_WEB_EXPOSURE_INCLUDE=prometheus,health
ANALYTICS_ACTUATOR_SECURITY_ENABLED=false
```

3.3.2. Servlet-Container

Dieser Abschnitt ist nur relevant, wenn Sie das Backend in einem Servlet-Container betreiben möchten.

Logging
Konfigurieren Sie das Logging in der Datei *WEB-INF/classes/logback-spring.xml*.

Eine allgemeine Beschreibung der XML Konfiguration für Logback finden Sie unter folgender Adresse: https://logback.qos.ch/manual/configuration.html.

Konfiguration
Um das Reporting zu konfigurieren, können Sie die WAR-Datei entweder entpacken und die Konfiguration unter *WEB-INF/classes/application-reporting.properties* vornehmen oder die Parameter über Umgebungsvariablen setzen. Weitere Konfigurationsdetails des Backends finden Sie in Abschnitt 2.4, „Konfiguration“.

3.4. Konfiguration

In diesem Abschnitt werden alle Konfigurationsparameter von Formcentric Analytics erläutert.

Wenn Sie die Anwendung innerhalb von FirstSpirit betreiben, können Sie die Konfiguration im Server Manager vornehmen.

analytics.backend.url
URL des bereits deployten Formcentric Analytics Backend

analytics.security.oauth2.clientSecret
Hier muss dasselbe Passwort hinterlegt werden, das bereits in der Backend-Konfiguration vergeben wurde.

3.4.1. Request-ID

Formcentric Analytis kann eine Request-ID aus den Http-Request-Headern entnehmen und ins Log ausgeben. Dies ermöglicht die Verfolgung eines Requests über mehrere Logeinträge hinweg. Formcentric Analytics setzt dazu eine Variable *REQ-ID* in den Mapped Diagnostic Context (MDC) des verwendeten Logging-Frameworks (siehe Abschnitt 3.3.1, „Spring-Boot“ bzw. Abschnitt 3.3.2, „Servlet-Container“).

Mit den folgenden Parametern können Sie diese Funktionalität konfigurieren.

analytics.http.requestId.enabled: Dieser Parameter ermöglicht es Ihnen, diese Funktion zu aktivieren (*true*) oder zu deaktivieren (*false*). Standardmäßig ist diese Funktionalität aktiviert.

analytics.http.requestId.headerName: Mit diesem Parameter können Sie den Namen des Http-Headers konfigurieren, aus dem Formcentric Analytics die Request-ID entnimmt. Standardmäßig ist der Wert *x-request-id*.

analytics.http.requestId.internalRequestIdPrefix: Wenn der Request keine Request-ID enthält, vergibt Formcentric Analytics automatisch eine Request-ID. Mit diesem Parameter können Sie den Präfix dieser automatisch generierten ID festlegen, um die ID im Log besser unterscheiden zu können.

analytics.http.requestId.externalRequestIdPrefix: Formcentric Analytics kann einen Präfix vor die externe Request-ID (die ID aus dem Request-Header) setzen, um im Log deutlich zu machen, dass diese ID nicht von Formcentric Analytics vergeben wurde.

3.4.2. Anbindung SSO / OpenId-Connect (Optional)

Um OIDC in Ihrer Reporting-Webapp zu nutzen, müssen neben der Backend-Konfiguration (siehe Abschnitt 2.4.5, „Anbindung SSO / OpenId-Connect (Optional)“) auch einige Einstellungen in der Reporting-Webapp vorgenommen werden.

analytics.oidc.clientId: Geben Sie hier den Identifier der Application an, unter der sich die Benutzer anmelden.

analytics.oidc.authorityUrl: Tragen Sie hier die URI Ihres Authorisierungs-Servers ein, die in Ihren JWTs unter *iss* steht. Für diese Einstellung muss Ihr Authorisierungs-Server einen Provider Configuration Endpunkt bereitstellen (siehe https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig). Dieser Endpunkt wird auch Authorization Server Metadata Endpunkt genannt (siehe https://datatracker.ietf.org/doc/html/rfc8414#section-3). Sollte dies nicht der Fall sein, wenden Sie sich bitte an unseren Support. Wir helfen Ihnen bei der Konfiguration.

analytics.oidc.scopes: Geben Sie die Scopes an, die vom OIDC-Server angefragt werden (mit Leerzeichen getrennt). Wenn diese Einstellung nicht vorgenommen wird, sind die Standardwerte *openid email profile*. Der Scope *openid* muss jedoch immer angegeben werden.

analytics.oidc.showLogout: Mit dieser Einstellung legen Sie fest, ob der Logout-Button im Reporting angezeigt wird. Der Standardwert ist *true*. Klickt der Benutzer auf den Button, erfolgt auch eine Abmeldung beim OIDC-Server. Wenn dies nicht gewünscht ist, können Sie den Button deaktivieren.

analytics.oidc.logoutView: Mit dieser Einstellung legen Sie fest, ob das Reporting nach dem Logout (über den Logout-Button, siehe oben) auf eine eigene Logout-Seite weiterleitet (*ANALYTICS*) oder ob der Benutzer direkt zur Login-Seite des OIDC-Servers weitergeleitet wird (*OIDC*). Auf der Analytics-Logout-Seite wird dem Benutzer nur ein Button angezeigt, der ihn zum Login des OIDC-Servers weiterführt. So kann der Benutzer sehen, dass er ausgeloggt wurde. Wenn dieser Parameter nicht gesetzt wird, ist der Standardwert *OIDC*, wodurch der Benutzer nach dem Logout direkt zur Login-Seite des OIDC-Servers gelangt.

3.4.3. Verschlüsselung von Konfigurationsparametern

In der Grundkonfiguration werden sensitive Daten im Klartext in Konfigurationsdateien gespeichert. Bei einem eventuellen Einbruch können so gültige Zugangsdaten entwendet werden. Aus diesem Grund haben Sie die Möglichkeit Passwörter auch verschlüsselt abzulegen. Die Passwörter werden in diesem Fall erst beim Start der Anwendung mit dem hinterlegten Verschlüsselungspasswort entschlüsselt. Das zur Verschlüsselung verwendete Passwort müssen Sie vor dem Start der Anwendung in einer Umgebungsvariable speichern. Standardmäßig wird hiefür die Umgebungsvariable *MWF_ENCRYPTION_PASSWORD* verwendet.

Um Werte zu ver- und entschlüsseln gibt es ein CommandLine Tool, das mit dem Befehl

```
mvn dependency:copy -Dartifact=com.formcentric:encryption-cli:2.0:jar
      -DoutputDirectory=.
```

bezogen werden kann. Bei einem Aufruf wird es entweder einen Wert ver- oder entschlüsseln.

```
java -jar encryption-cli-1.0.jar -p 'encryptionPassword'
      -e 'toEncrypt'
```

Falls es *ohne* Parameter gestartet wird, wird es nach dem Passwort und den zu ver- bzw. entschlüsselnden Werten fragen. Bitte beachten Sie, dass die Parameter in einfachen Anführungszeichen angegeben werden müssen. Es kann mit den folgenden Parametern gestartet werden:

Parameter	**Beschreibung**
-p oder --encryption-Password	Um das Passwort anzugeben das zur Ver- oder Entschlüsselung verwendet wird.
-d oder --decrypt	Um den darauf folgenden Wert zu entschlüsseln.
-e oder --encrypt	Um den darauf folgenden Wert zu verschlüsseln.
-? oder --help	Um die Hilfe zu dem Tool anzuzeigen.