openKonsequenz - Architecture of the module *Contact Base Data*

Frank Dietrich, Simon Reis

Table of Contents

Introduction and Goals . . . 2
- Requirements Overview . . . 2
- Quality Goals . . . 2
- Stakeholders . . . 3

Architecture Constraints . . . 4
- Technical Constraints . . . 4
- Technical Dependencies . . . 5

System Scope and Context . . . 9
- Business Context . . . 9
- Technical Context . . . 9
- Solution Strategy . . . 9

Building Block View . . . 10
- Whitebox Overall System . . . 10
- Level 2 . . . 11

Runtime view . . . 13
- Login / authentication . . . 13
- Deployment of the application components . . . 13
- CI- and CD-Components . . . 16
- Continuous deployment . . . 16

Design decisions . . . 18

Risks and Technical Debts . . . 19

Glossary . . . 20

This documentation is based on the ARC42-Template (v7.0):

Introduction and Goals

Requirements Overview

Many user modules of an openKONSEQUENZ installion need contact data for their daily business. Furthermore they have to fulfil the regulatory requirement of the General Data Protection Regulation (GDPR).

This core module *Contact Base Data* provides a central component for managing contact datas including the crucial functionality of GDPR.

The full requirements of the module *Contact Base Data* (in German: Modul *Kontaktstammdaten*) is described in the document

- "Anforderungsspezifikation Modul Kontaktstammdaten" version 1.2 / 07-11-2019.

Quality Goals

The module *Contact Base Data* represents a core module that is based on the architecture platform of openKONSEQUENZ. The main quality goals of the platform are:

- **Flexibility** The reference platform shall grant that different systems and modules from different vendors/developers can interact and interoperate, and may be exchanged or recombined.
- **Availability** All platform modules that are running on the platform can only be as available as the platform same for user modules that are based on platform modules.
- **Maintainability** (and testability as part of maintainability) The platform and its platform modules shall be used longer than 15 years.
- **Integration performance** New implemented functionality of oK's own modules and external modules shall be included fast / automatically.
- **Security** The platform and its modules need to underly security-by-design

The main quality goals of the core module Contact Base Data are:

- **Functionality** The core module must fulfil the functional requirements mentioned in the section before
- **Ergonomics** The web interface must be realized according to oK-GUI-styleguide.
- **Good documentation** (i.e. code and architecture documentation) makes code changes easier and automatic tests facilitate rigorous verification.
- **Modifiability** (and testability as part of modifiability)
- **Integration performance** The core module's integration into different production environments has to be easy.

The following documents contain the quality goals in detail:

- Architecture Committee Handbook v1.6.0 from 10-07-2019
- Quality Committee Handbook v2.0.1 from 15-10-2018

The architecture is based on the AC-Handbook. The quality demands are described in the QC-Handbook. Both specifications were fully complied with in the project, so that a high quality is given.

The code quality regarding static code analysis and unit test code coverage on the backend and frontend sides are ensured by the use of sonarqube. The rule set and the qualtity gate are defined by the default, the so called "sonar way".

The module *Contact Base Data* is part of the Eclipse project *Eclipse openK Core Modules*. This project bases on the Eclipse Public Licence 2.0.

Stakeholders

Table 1. Stakeholders

Role/Name	Contact	Expectations
Product Owner (represents the Distribution System Operators)	Gordon Pickfort, Rainer Fuhrmann	The software must fulfil their functional and nonfunctional requirements.
Module Developer	Michel Alessandrini, Jonas Tewolde, Frank Dietrich	All relevant business and technical information must be available for implementing the software.
External Reviewer (represents the AC/QC)	n.n.	The software and the documentation is realized according to the Quality and Architecture Handbook of openKONSEQUENZ.
External Reviewer (represents the Eclipse-Requirements)	n.n.	The software is licensed under the EPL 2.0. It must be validated that all requirements are fulfilled.
System Integrator	n.n.	A documentation for the integration of the module in the DSO specific environments must be available.

Architecture Constraints

The main architecture constraints are:

- **Public License** The module must be available under the “Eclipse Public License 2.0”.
- **Standardization** The module must use the reference platform.
- **Availability** The source code of the module must be accessible to any interested person/company.

Therefore the project is published under the following repositories:

- https://git.eclipse.org/r/openk-usermodules/org.eclipse.openk-usermodules.contactBaseData.backend
- https://git.eclipse.org/r/openk-usermodules/org.eclipse.openk-usermodules.contactBaseData.frontend

Technical Constraints

The following technical constraints are given:

Table 2. Technical Contraints

Component	Constraints
Base components of the reference platform	* Application Server Tomcat * JPA EclipseLink * Database PostgreSQL
Enterprise service bus	* ESB Talend * Communication via RESTful Webservices
Programming language frontend	* Angular * Bootstrap * jQuery * REST/JSON Interfaces
GUI design	* According to oK-GUI-Styleguide
Java QA environment	* Sonarqube 5.6.6
Programming language	* Backend: Java 1.8 * Frontend: Angular 7+ (Javascript, Typescript, HTML5, CSS3)
IDE	* Not restricted (Eclipse, IntelliJ, Microsoft Developer Studio, Microsoft Visual Code ...)
Build system	* Backend: Maven * Frontend: NodeJS + Angular/cli
Libraries, frameworks, components	* Used Libraries/Frameworks have to be compatible to the Eclipse Public License
Architecture Documentation	* According ARC42-Template

Technical Dependencies

Modules

The following modules are required to use the *Contact Base Data*:

Table 3. Modules

Name of the module	Purpose	Status of the module
Auth&Auth	Authentification and Authorization	available

Libraries

The following libraries are used:

Table 4. Libraries

Name of the library	Version	Artefact-id	Usage	License	Tier
org.springframework.boot.spring-boot-starter-parent	2.2.1.RELEASE			Apache License 2.0	Backend
org.springframework.boot.spring-boot-starter-data-jpa	2.2.1.RELEASE			Apache License 2.0	Backend
org.springframework.spring-boot-starter-data-ldap	2.2.1.RELEASE			Apache License 2.0	Backend
org.springframework.boot.spring-boot-starter-oauth2-client	2.2.1.RELEASE			Apache License 2.0	Backend
org.springframework.boot.spring-boot-starter-security	2.2.1.RELEASE			Apache License 2.0	Backend
org.springframework.boot.spring-boot-starter-web	2.2.1.RELEASE			Apache License 2.0	Backend
org.flywaydb.flyway-core	6.0.8			Apache License 2.0	Backend

Name of the library	Version	Artefact-id	Usage	License	Tier
org.springframework.cloud.spring-cloud-starter-openfeign	2.2.0.RELEASE			Apache License 2.0	Backend
org.springframework.cloud.spring-cloud-starter-netflix-ribbon	2.2.0.RELEASE			Apache License 2.0	Backend
org.keycloak.keycloak-core	3.4.2_Final			Apache License 2.0	Backend
org.postgresql.postgresql	42.2.8			New BSD License	Backend
org.projectlombok.lombok	1.18.10			MIT	Backend
org.mapstruct.mapstruct-processor	1.2.0.Final			Apache License 2.0	Backend
io.jsonwebtoken.jjwt	0.9.1			Apache License 2.0	Backend
io.springfox.springfox-swagger2	2.9.2			Apache License 2.0	Backend
io.springfox.springfox-swagger-ui	2.9.2			Apache License 2.0	Backend
org.springframework.boot.spring-boot-starter-test	2.2.1.RELEASE			Apache License 2.0	Backend
org.springframework.security.spring-security-test	5.2.1.RELEASE			Apache License 2.0	Backend
org.powermock.powermock-reflect	2.0.0			Apache License 2.0	Backend
com.h2database.h2	1.4.200			EPL	Backend

Name of the library	Version	Artefact-id	Usage	License	Tier
org.springframework.cloud.spring-cloud-dependencies	Hoxton.RELEASE			Apache License 2.0	Backend
org.springframework.boot.spring-boot-maven-plugin	2.2.1.RELEASE			Apache License 2.0	Backend
org.jacoco.jacoco-maven-plugin	0.7.9			EPL 2.0	Backend
org.sonarsource.scanner.maven.sonar-maven-plugin	3.2			LGPL 3.0	Backend
org.asciidoctor.asciidoctor-maven-plugin	1.5.3			Apache License 2.0	Backend
org.jruby.jruby-complete	9.0.0.0			EPL 2.0	Backend
org.asciidoctor.asciidoctorj-pdf	1.5.0-alpha.11			Apache 2.0	Backend
org.asciidoctor.asciidoctorj	1.5.4			Apache 2.0	Backend
org.asciidoctor.asciidoctorj-pdf	1.5.0-alpha.11			Apache 2.0	Backend
org.asciidoctor.asciidoctorj-diagram	1.5.4.1			Apache 2.0	Backend
Angular Font Awesome	3.1.2			MIT License	Frontend
@auth0/angular-jwt	3.0.1			MIT License	Frontend
font-awesome	4.7.0			MIT License	Frontend
@ngrx/core	1.2.0			MIT License	Frontend
@ngrx/effects	8-2-0			MIT License	Frontend
@ngrx/store	8.3.0			MIT License	Frontend
@ngrx/store-devtools	8.2.0			MIT License	Frontend

Name of the library	Version	Artefact-id	Usage	License	Tier
@ngx-translate/core	11.0.1			MIT License	Frontend
@ngx-translate/http-loader	4.0.0			MIT License	Frontend
ag-grid-angular	21.2.1			MIT License	Frontend
ag-grid-community	21.2.1			MIT License	Frontend
angular2-notifications	2.0.0			MIT License	Frontend
bootstrap	4.4.1			MIT License	Frontend
jquery	3.4.1			MIT License	Frontend
classlist.js	1.1.20150312			MIT License	Frontend
core-js	3.2.1			MIT License	Frontend
moment	2.24.0			MIT License	Frontend
ng2-popover	0.0.14			MIT License	Frontend
ngrx-forms	5.2.1			MIT License	Frontend
npm-install-peers	1.2.1			MIT License	Frontend
reselect	4.0.0			MIT License	Frontend
rxjs	6.5.3			MIT License	Frontend
rxjs-compat	6.5.4			MIT License	Frontend
ts-helpers	1.1.2			MIT License	Frontend
tslib	1.10.0			MIT License	Frontend
web-animations-js	2.3.2			MIT License	Frontend
zone.js	0.10.1			MIT License	Frontend
@swimlane/ngx-datatable	15.0.2			MIT License	Frontend
puppeteer	2.0.0			MIT License	Frontend
ngx-toastr	11.2.1			MIT License	Frontend
popper.js	1.16.0			MIT License	Frontend
@ng-bootstrap	5.1.5			MIT License	Frontend

System Scope and Context

Business Context

The core module *Contact Base Data* communicates via Restful Webservices with the follwowing modules:

- **Core Module *Auth & Auth*** The *Contact Base Data* can only be used by authorized users. Therefore, it is essential to invoke the module *Auth & Auth* for authorization and authentication purposes.

Technical Context

The following aspects have to be taken into account for external communication of the module *Contact Base Data*:

- RESTful web services are used as interface-technology.
- Each external interface (interfaces between modules or external systems) has to be documented.
- Dependencies of modules to services realized by other modules have to be specified and documented explicitly.

The interfaces of the module *Contact Base Data* are described in the interface documentation.

Solution Strategy

The module *Contact Base Data* is based on a three-tier architecture:

1. **Frontend** - The GUI is implemented as a web-frontend with rich-client functionality.
2. **Backend** - The business functionalities are implemented in the backend tier. It provides the business functions via RESTful Webservices.
3. **Database** - The database stores all module specific data.

Building Block View

Whitebox Overall System

The module *Contact Base Data* contains two components (see figure 2):

1. **UI** - Represents the graphical user interface and consumes the services from the business logic component via RESTful webservices.
2. **Business Logic** - Realizes the business functionality and the data storage of the module. The module itself is split up into several components due to the requirement to use microservices.

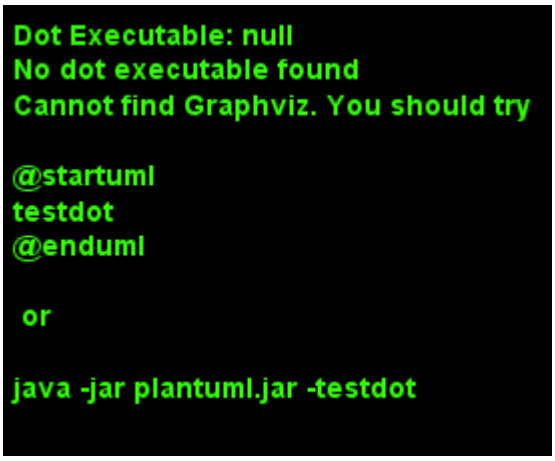

Figure 1. Module components

The communication between WebBrowser and Apache Tomcat is established via HTTP/HTTPS. ApacheTomcat is connected to the data source (PostgresDBMS) via TCP/IP.

contactBaseDataFE

This component implements the presentation logic for the **contact-base-data**-module using the **Angular**-TypeScript framework. The Frontend is a so called **Single Page Application** (SPA) because it behaves like a single HTML-page.

contact-base-data.jar (backend tier)

This component implements the business functionality of the contact base data. And it provides services, that the contactBaseDataFE can use the functions in the frontend.

The "spring boot/spring cloud" framework is used to implement this application.

ContactBaseDataDev-DB (Database tier)

This component stores the data of the contact base data. It provides an interface to the contact-base-data.jar to create or change data in the database.

The ContactBaseDataDev-DB runs on a Postgres DBMS. (The decision to use the Postgres DBMS was made by the openKONSEQUENZ architecture committee)

Level 2

ContactBaseDataFE (frontend tier)

The frontend component implements the concept of a single-page application (SPA). The framework used is Angular 8.

It divides the contactBaseDataFE into three layers:

1. **Components** - The components (pages, lists, dialogs, common comp.) represent the presentation layer and the control layer. A component contains the control logic (.ts-file), an HTML-fragment as presentation description (.html-file) and a style definition (.css-file).
2. **Services** - The service component communicates with the interfaces of the backend via HTTP requests by using the model component.
3. **Model** - The model corresponds to the view-model of the backend tier.

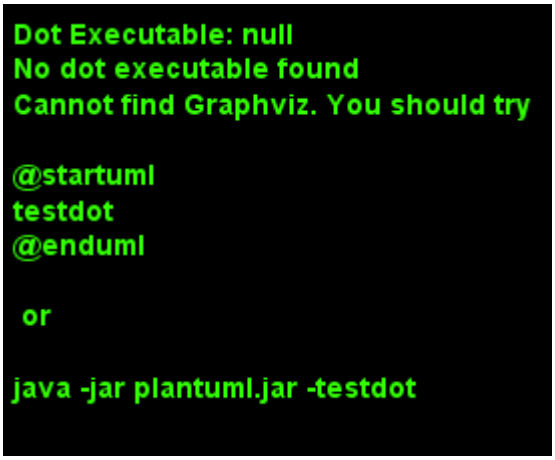

Figure 2. Frontend tier

contact-base-data.jar (backend tier)

The backend tier contains five components which can be summarized in three layers:

1. **Presentation layer** - Represented by
   a. REST-Srv
   b. View model/DTO
2. **Controller layer** - Represented by
   a. Controller
   b. Service
3. **Model layer** - Represented by
   a. Repository
   b. Model

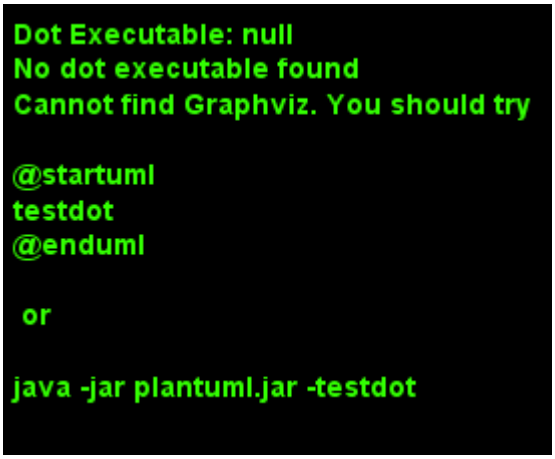

Figure 3. Backend tier

ContactBaseData-DB (database tier)

The ContactBaseData-DB is realized as a relational database system.

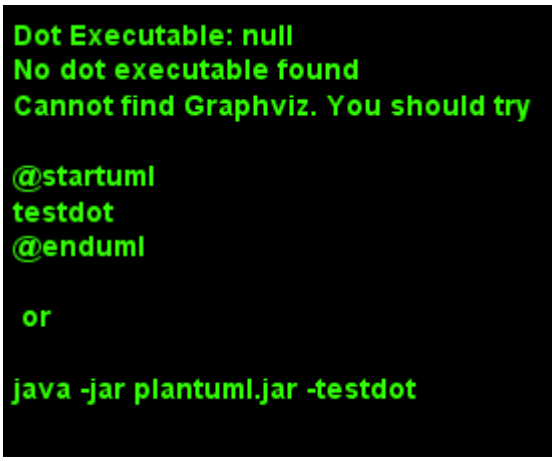

Figure 4. Database tier

Program Configuration

Runtime view

Login / authentication

There is no login page, since the openK-Portal-Application is responsible for authentication and the whole SSO (single sign on) process. Therefore the application has to be started by providing a valid authentication token. This token is a JWT (JSON Web Token).

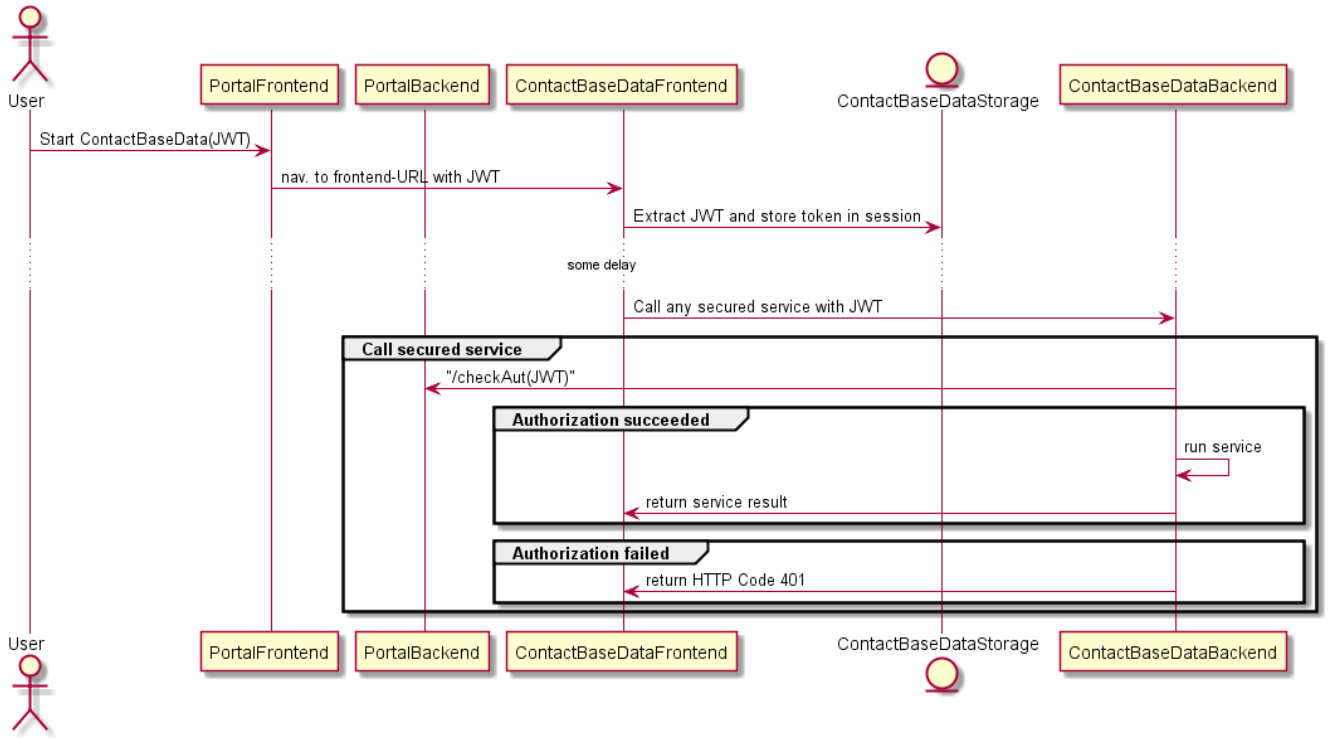

Figure 5. contactBaseData application is called by the **portal** *application. The User is already logged in*

Deployment of the application components

Deployment of the frontend

See "howtoRun.pdf"

Deployment of the backend

See "howtoRun.pdf"

Deployment of the database

See "howtoRun.pdf"

The component "Flyway" is used to make to distribute structural or content related changes to the database.

The database is built out of the scripts in the directory "db/migrations". Every SQL script contains

the complete db script for the contact base data database (in different versions). The highest version number indicates the currently valid script.

Configuration of the system

DB based configuration

See "howtoRun.pdf"

Configuration of the contact base data backend

The backend service is configured in the * .yaml files, which are located in the JAR file.

This yml-file can be divided into different configuration profiles. When starting the backend-service one has the possibility to specify the active profile.

- **spring**:
  - **configuration**: Section for the database connection
  - **flyway**:
    - **enabled**: (true or false) If enabled=true then the database migrations will be performed automatically when starting the application (this parameter should normally be set to "false")
  - **ldap**:
    - **base**: The base LDAP path
    - **port**: The LDAP server port
    - **username**: Admin user of your LDAP
    - **password**: Admin password
    - **urls**: The URL of the LDAP server should be in the format ldap://myserver.example.com:10389. For SSL access, use the ldaps protocol and the appropriate port, e.g. ldaps://myserver.example.com:636
- **ldap-sync**:
  - **attribute-mapping**:
    - [not changeable variable in contact base data modul]: [attribute field name in your LDAP to be mapped]. All possible mappings are:

```
uid: uid
fullname: cn
lastname: sn
firstname: givenname
title: title
mail: mail
department: department
telephone-number: phone
```

- **db-id-mapping**:
  - **telephone-number-id**: the primary ID for "telephone number" row in table REF_COMMUNICATION_TYPE. (Default 1) If set to -1 ldapmapping is disable for "telephone number"
  - **mail-id**: the primary ID for "mail" row in table REF_COMMUNICATION_TYPE. (Default 2) If set to -1 ldapmapping is disable for "mail"
- **scheduling**:
  - **enabled**: (true or false) Switches LDAP synchronisation on/off
  - **cron-expression**: Cron Trigger Tutorial
    Spring Cron only takes 5 parameters not 6, the year is exlcuded!
    Examples: **/10 * * * * ** = every 10 seconds, *0 0 */3 ? * ** = every 3 hours, *0 0 0 * * ?* = every day at midnight.
    - **authnauth-sync**:
- **attribute-mapping**:
  - **lastname**: (true or false) Switches AuthNAuth synchronisation of field Lastname on/off
  - **firstname**: (true or false) Switches AuthNAuth synchronisation of field Firstname on/off
- **scheduling**:
  - **enabled**: (true or false) Switches AuthNAuth synchronisation on/off
  - **cron-expression**: Cron Trigger Tutorial
    Spring Cron only takes 5 parameters not 6, the year is exlcuded!
    Examples: **/10 * * * * ** = every 10 seconds, *0 0 */3 ? * ** = every 3 hours, *0 0 0 * * ?* = every day at midnight.
- **technical-username**: A technical user from the AuthNAuth modul doesn't have to be an admin
- **technical-userpassword**: Technical user password
  - **server**:
- **max-http-header-size**: Maximum size for the http-headers
  - **jwt**:
- **tokenHeader**: Name of the http-header which carries the authentication-token. (should be

"Authorization")

- **useStaticJwt**: If set to "true" then the backend will use **jwt.staticJwt** as Authorization-token. (This won't work for calls to other modules like the Auth'n'Auth-Modul, because the token will be out of date)
  - **services**:
- **authNAuth**:
  - **name**: authNAuthService
    - **authNAuthService**:
- **ribbon**:
  - **listOfServers**: Here one can configure the base URL to the Auth'n'Auth-service Example: http://entopkon:8880
    - **feign**:
- **client**:
  - **config**:
    - **default**:
      - **connectTimeout**: (Default 60000) Connection timeout for the REST-Calls (in ms).
      - **readTimeout**: (Default 60000) Read timeout for the REST-Calls (in ms).
        - **cors**:
- **corsEnabled**: (true or false) Cross-Origin Resource Sharing on/off

CI- and CD-Components

GIT-Repository

Backend: https://git.eclipse.org/c/openk-coremodules/org.eclipse.openk-coremodules.contactBaseData.backend.git/

Frontend: https://git.eclipse.org/c/openk-coremodules/org.eclipse.openk-coremodules.contactBaseData.frontend.git/

Continuous deployment

The continuous deployment is realized on two platforms:

- the development platform (Dev-Environment)
- the quality platform (Q-Environment)

The automatic deployment on both of the environments is directly linked to the branches on the

GIT-repositories:

1. "SNAPSHOT" or "DEVELOP"
2. "MASTER" or "TRUNC"

The running development is exclusively made on the snapshot-branch. Every time a developer checks in (pushes) code to the repository, an automatic build starts on the hudson ci-server. If the snapshot-build is successful, then the result of that build is directly deployed on the dev-environment.

At the end of a scrum sprint or when a big user story is realized, all code changes are merged from the **SNAPSHOT**-Branch to the **TRUNC**. This automatically triggers the build and the deployment on the Q-environment.

Design decisions

All architecture decisions are based on the Architecture Committee Handbook. There are no deviations.

Risks and Technical Debts

(Currently there aren’t any known issues)

Glossary

Table 5. Abbreviations and glossary terms

Short	Long	German	Description
AC	Architecture Committee	Architektur-Komittee	Gives framework and constraints according to architecture for oK projects.
CNCU	Central Network Control Unit		
DAO	Data Access Objects		
DTO	Data Transfer Object		
DSO	Distribution System Operator	Verteilnetzbetreiber (VNB)	Manages the distribution network for energy, gas or water.
EPL	Eclipse Public License		Underlying license model for Eclipse projects like contact-base-data@openK
ESB	Enterprise Service Bus		Central instance to exchange data to overcome point-to-point connections.
oK	openKONSEQUENZ	openKONSEQUENZ	Name of the consortium of DSOs
QC	Quality Committee	Qualitätskomitee	Gives framework and constraints according to quality for oK projects.
SCADA	Supervisory Control and Data Acquisition	Netzleitsystem	System, that allows DSOs to view/control actual parameters of their power grid.