

A close-up, low-angle shot of a field of golden wheat or grain, with the stalks reaching upwards and slightly out of focus in the background, creating a warm, textured, and bokeh-filled background.

LF

NETWORKING

LFN Developer & Testing Forum



LFN Developer & Testing Forum

Deutsche Telekom AG TNAP Portal

Portal Technology + Demo

Georg Schweflinghaus, Andreas Geißler

AGENDA

- 01 Intro**
Why we do what we do?
Who we are?
- 02 Portal Development**
How do we do it?
- 03 Demo**
Where are we right now?
- 04 Issues**
How to cope?
- 05 Outlook**
What is planned next?

01 Intro - Why we do what we do

The conditions for our work are:

- Deutsche Telekom (DT) is trialing ONAP and ORAN
- The first real world use case for the ONAP platform is managing the Open Radio Access Network (O-RAN).
- ONAP already features various web applications.

In this context the goal is to:

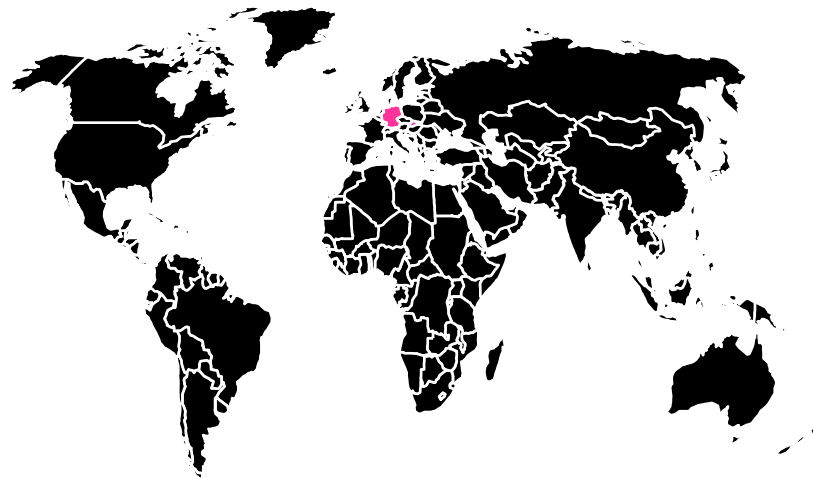
- fill gaps in the User Experience and functionality provided by standard ONAP components
- provide a twist towards radio engineering so that DT radio staff recognize their RAN domain, and we satisfy their requirements
- still remain generic to cater for new use cases while adapting the experience to the RAN domain

01 Intro - Who is we?

DT Portal team is one scrum team
inside the TNAP project

As of today:

- developers
- test automation engineers



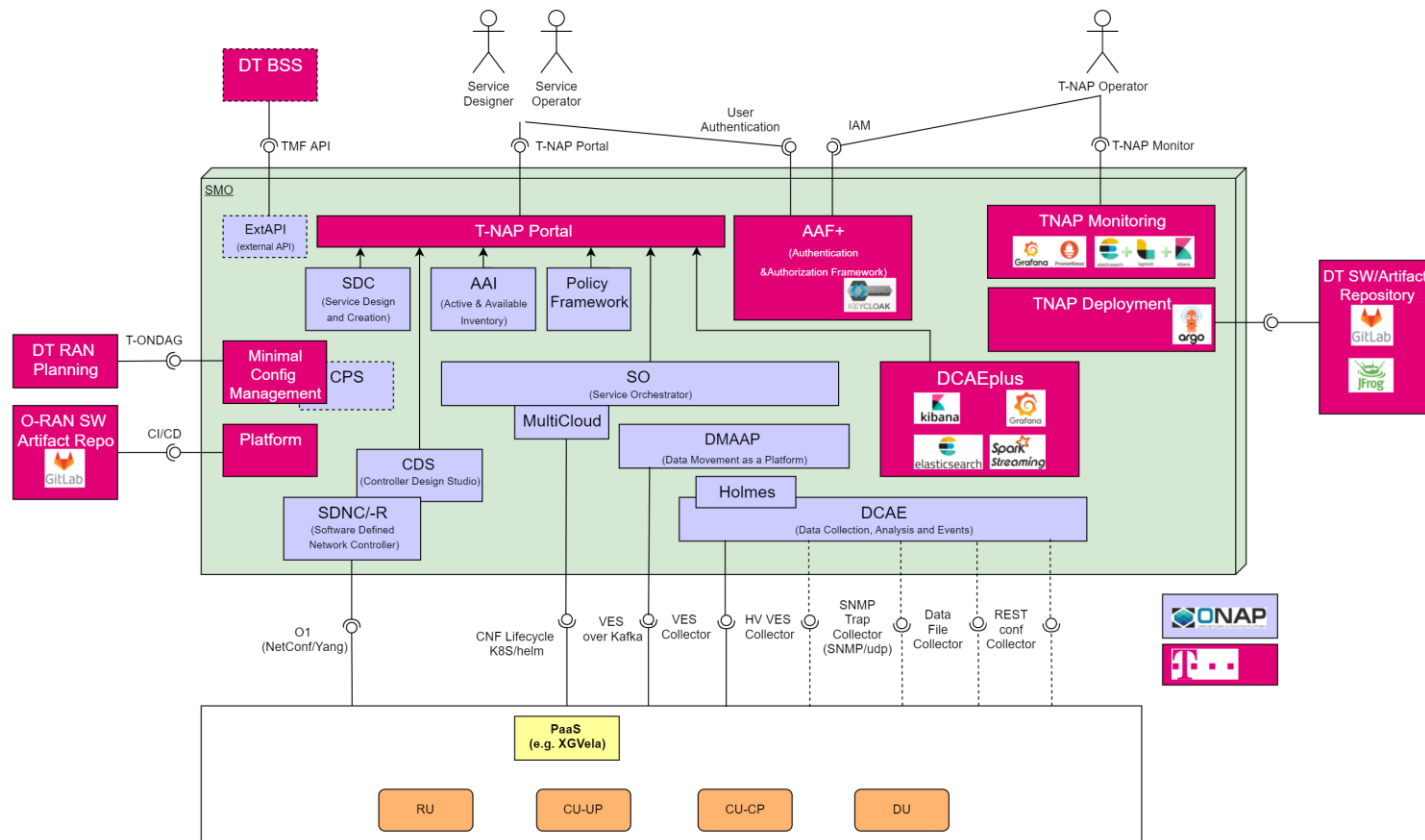
Slovakia: Kosice

Germany: Münster, Bonn

02 Portal Development – T-NAP architecture

Portal Focus:

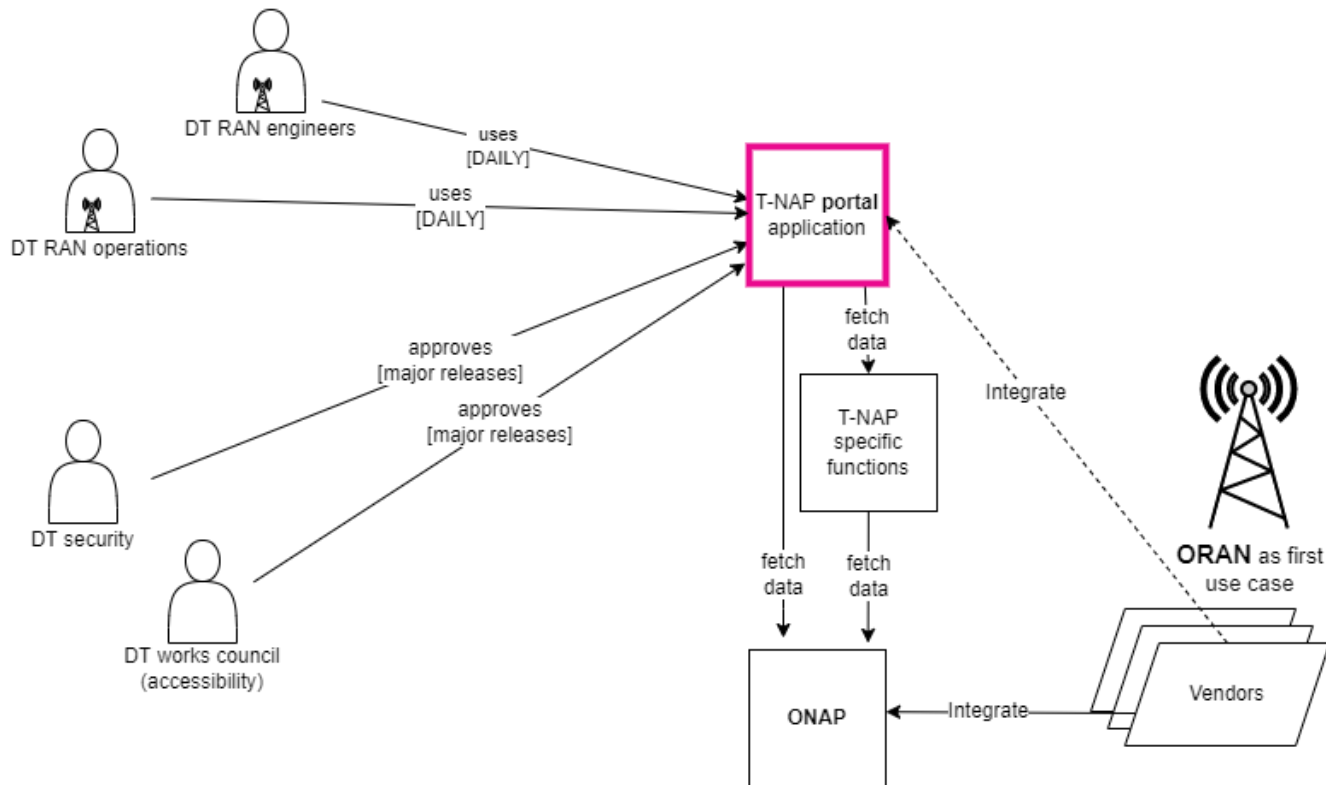
- **Role Based Access**
- **UI Application launch**
(ONAP + TNAP)
- **Service Instantiation**
(VID replacement)
- **Service Inventory**
(AAI topology view)
- **Service Monitoring**
(DCAE+)
- **Business Mgmt UI (planned)**
(Customer, LoB, Project...)
- **Service Infra UI (planned)**
(ESR UI replacement)
- **Usecase specific UIs**



02 Portal Development - Context

Portal is:

- Accessible
- Secure
- Manage ORAN as first use case
- Leveraging ORAN promise to onboard several vendors



02 Portal Development – High Level Concepts

Layered architecture



Frontend

- ✓ Developed using Angular
- ✓ Provides Single Page Application
- ✓ Communicates with Backend via REST requests



Backend

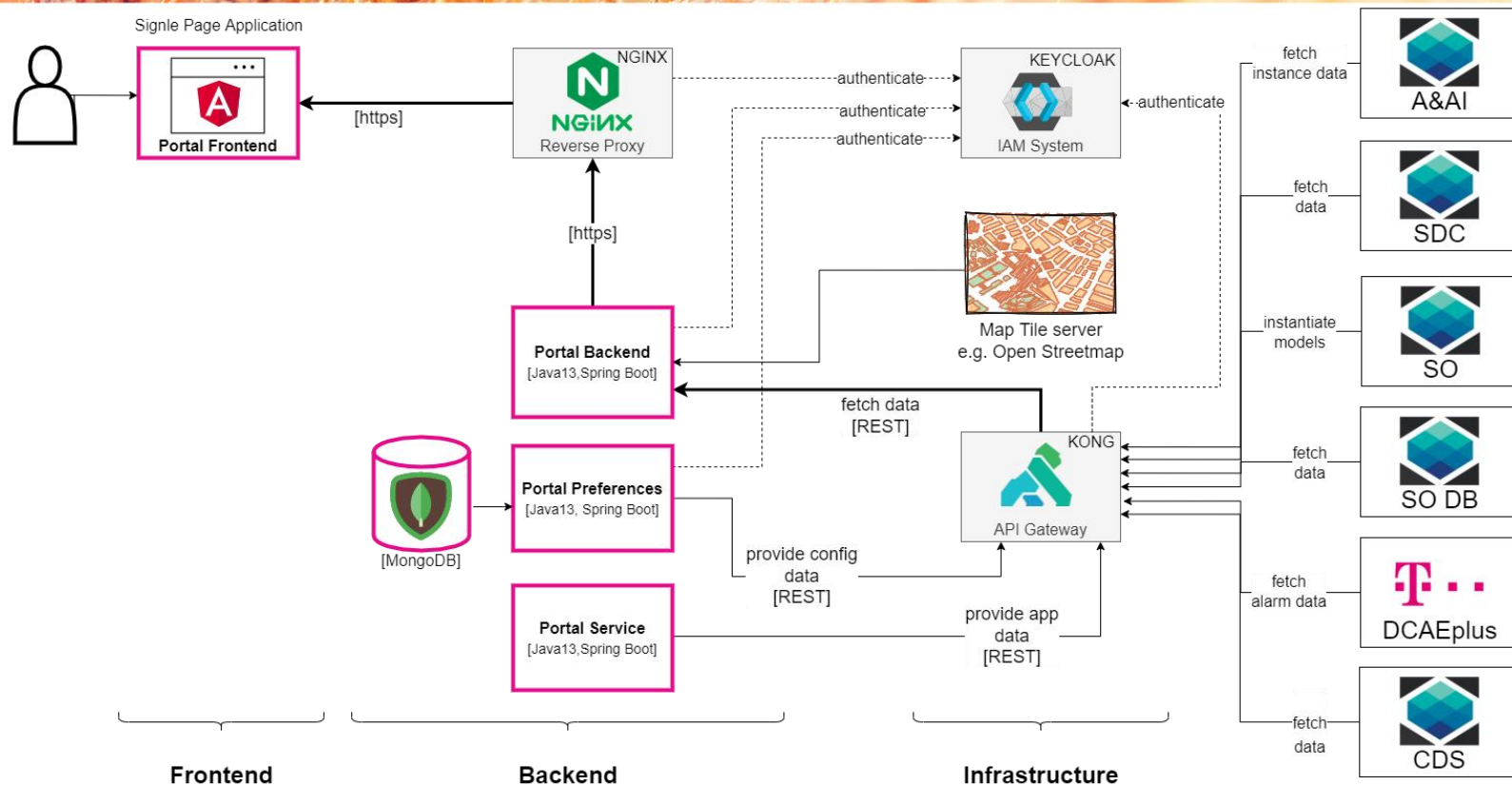
- ✓ Developed using Java 11 + Spring Boot
- ✓ Microservice based
- ✓ Reduces complexity of southbound APIs for Frontend



Southbound

- ✓ ONAP services: SDC, A&AI, ...
- ✓ Infrastructure services: API Gateway, Authentication, Maps

02 Portal Development – Detailed View



02 Portal Development - How we work!



Code Quality and Code Security



Mocking APIs with WireMock - <http://wiremock.org/>)



Generate clients, servers, and documentation from OpenAPI documents
- <https://openapi-generator.tech>



End to End test with: Allure- <http://allure.qatools.ru/>)

Serenity <https://serenity-bdd.info/what-is-serenity/>

02 Portal Development – Automatic Testing

End to End testing

Is achieved with a combination of serenity and Allure

The screenshot displays the Allure test results interface. On the left is a sidebar with navigation links: Overview, Categories, Suites, Graphs, Timeline, Behaviors, and Packages. The main area is titled 'Suites' and shows a list of test suites with columns for order, name, duration, status, and a count. The status bar at the top indicates 7 failed, 9 passed, 133 skipped, and 0 pending tests. The list includes suites like 'Alarm management - Column Ordering', 'Alarm management - Detail', and 'Alarm Management - List'. The 'Alarm Management - List' suite is expanded, showing individual test cases with their IDs, descriptions, and durations. For example, test #10 'Alarm can be successfully acknowledged in CLEARED tab' passed in 46s 452ms. On the right, the 'Description' and 'Execution' sections are visible. The description states: 'As an operations engineer I want an overview on currently active alarms and warnings so that I can assess the overall status of the network I have to operate.' The execution section shows a 'Set up' step followed by a 'Test body' with multiple steps, including user login, button clicks, and assertions. A screenshot of the application interface is also shown at the bottom right.

Suites

order	name	duration	status	count
	Alarm management - Column Ordering		2	2
	Alarm management - Detail		4	4
	Alarm Management - List		2	9
8	Alarm can be successfully acknowledged in ACTIVE tab	51s 745ms	2	2
10	Alarm can be successfully acknowledged in CLEARED tab	46s 452ms	2	2
7	Alarm can be successfully cleared	36s 632ms	2	2
9	Alarm can be successfully unacknowledged in ACTIVE tab	50s 605ms	2	2
11	Alarm can be successfully unacknowledged in CLEARED tab	35s 935ms	2	2
3	Alarm List can be refreshed automatically after 1 minute	1 minute, Auto refresh ev... 14s 787ms	2	2
4	Alarm List can be refreshed automatically after 5 minutes	5 minutes, Auto refresh ev... 12s 384ms	2	2
2	Alarm List can be refreshed manually	18s 931ms	2	2
5	Auto refresh button on Alarm list page can be turn OFF	12s 468ms	2	2
1	List of Alarms can be shown	12s 219ms	2	2
6	List of Alarms has visible badges	13s 516ms	2	2
	Alarm Management - List - Filter		2	7
	Alarm Management - List - Search		2	7
	Alarm Management - List - Sort		2	7
	App Starter - Tiles		2	4
	Cell Site Map		2	7
	Dashboard - KPI Graphs		2	4
	Dashboard - Menu		2	7

Description

As an operations engineer I want an overview on currently active alarms and warnings so that I can assess the overall status of the network I have to operate.

Execution

Set up

Test body

- Given User is logged in UI as TEST_ADMIN 13 sub-steps 6s 529ms
- When User clicks on ALARM_MANAGEMENT button in main menu 2 sub-steps 185ms
- Then User can see ALARM_MANAGEMENT_PAGE 7 sub-steps 4s 547ms
- And User clicks on CLEARED TAB on Alarm Management List 23 sub-steps 5s 523ms
- And User clicks on first UNACKNOWLEDGED Alarm in list and saves Alarm's unique id 9 sub-steps 7s 446ms
- And User searches for a specific Alarm by unique id using SEARCH 2 sub-steps 429ms
- When List contains 1 Items 2 sub-steps, 1 attachment 21s 660ms
 - By.xpath: //div[contains(@class, 'loading-spinner')] should be(visible, 3 s.) 931ms
 - By.xpath: //div[contains(@class, 'loading-spinner')] should be(hidden) 1 attachment 20s 723ms
- Screenshot 127.2 KB

03 Demo – The Real Thing

Functions shown in the demo

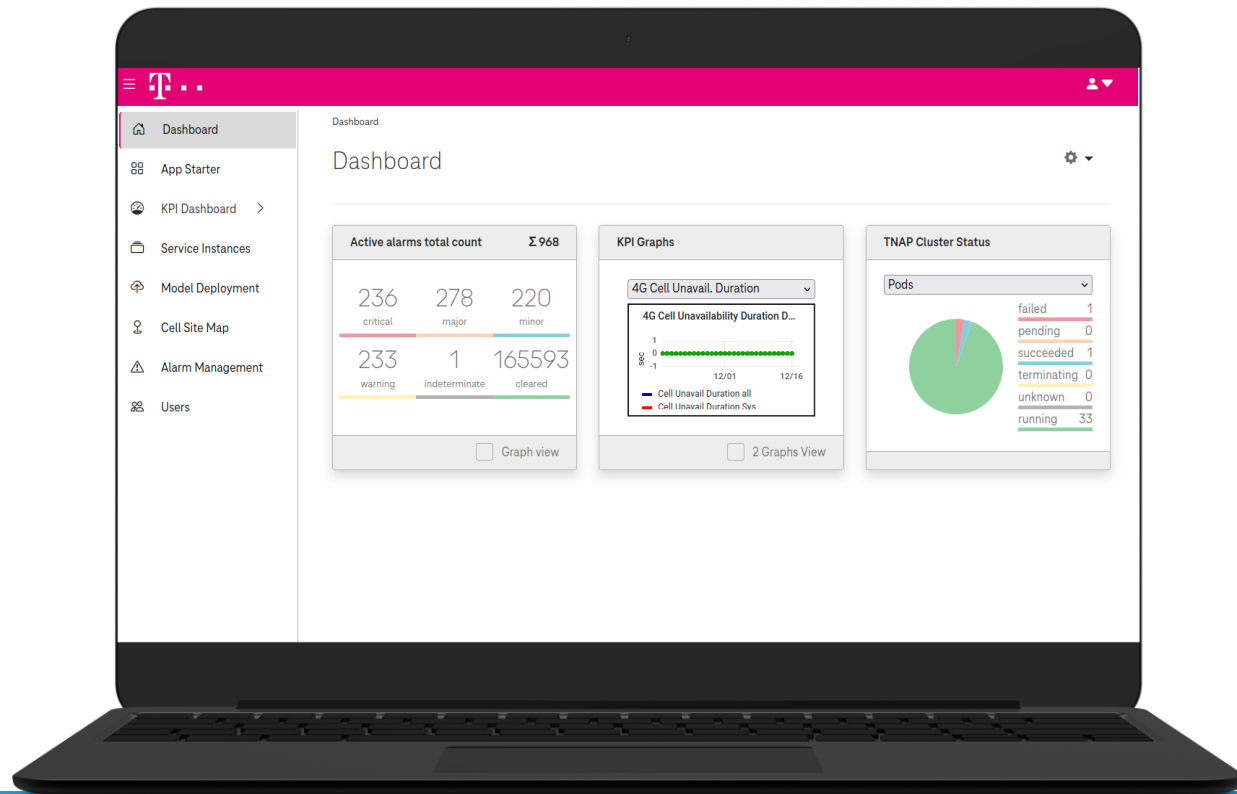
01 App Starter

02 Model Deployment

03 Instance View

04 Topology View

05 Alarm View



Size of ONAP deployment

- Parallel development with several teams requires multiple setups e.g. for independent automatic tests
- ONAP has extensive resource requirements and running several clusters is thereby expensive

Incomplete API descriptions

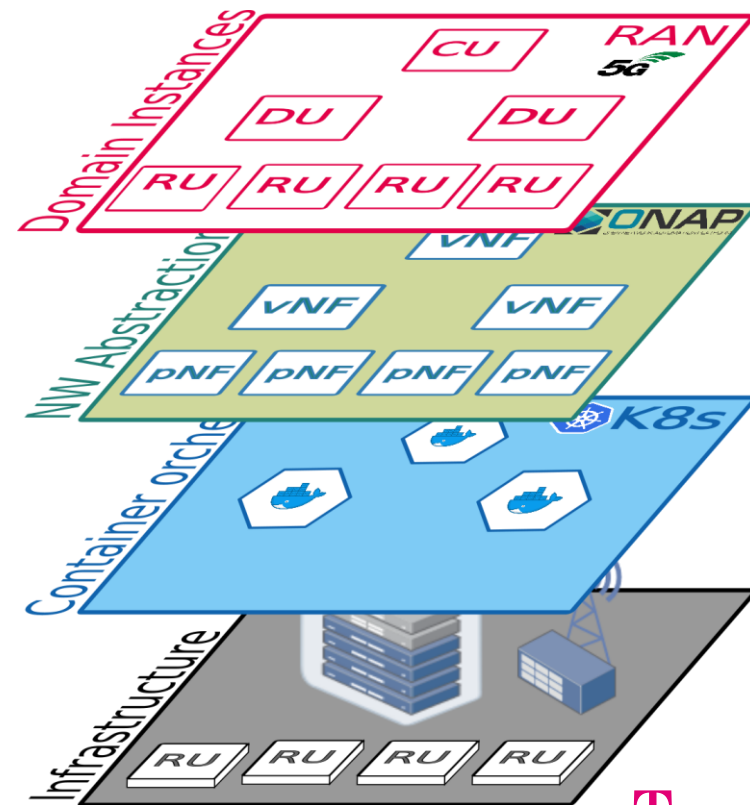
- APIs provided via „Open API File“ are partially not described, described wrong or even not existing
 - As one example, for SO we had to write/reverse engineer our own file to get instantiation working
 - Every cloud has a silver lining, the approach „Use the source Luke!“ was working for us.

04 Issues

Teach ONAP to NW operation or

customize ONAP to mimic domain of NW operation

- Development teams face the challenge of mastering: SW Domain + ONAP Domain + RAN Domain
- Users are experts in their domain but usually not in the ONAP domain



05 Outlook – What is planned next?

“Just saying, just some keywords”

Specific for the portal:

- Day 2 Network Management e.g.:
 - Radio Site/Cell handling
 - RAN monitoring
 - RAN trouble shooting
- Use domain APIs that are built on top of NetConf
- Treeview for NW discovery
- Business context management (Customer, Project, LineOfBusiness,...)
- Service Infrastructure Management (Cloud region, ESR GUI replacement)

General:

- Midterm prepare for contribution of portal to ONAP
- ORAN a first domain handled as plug-in/module