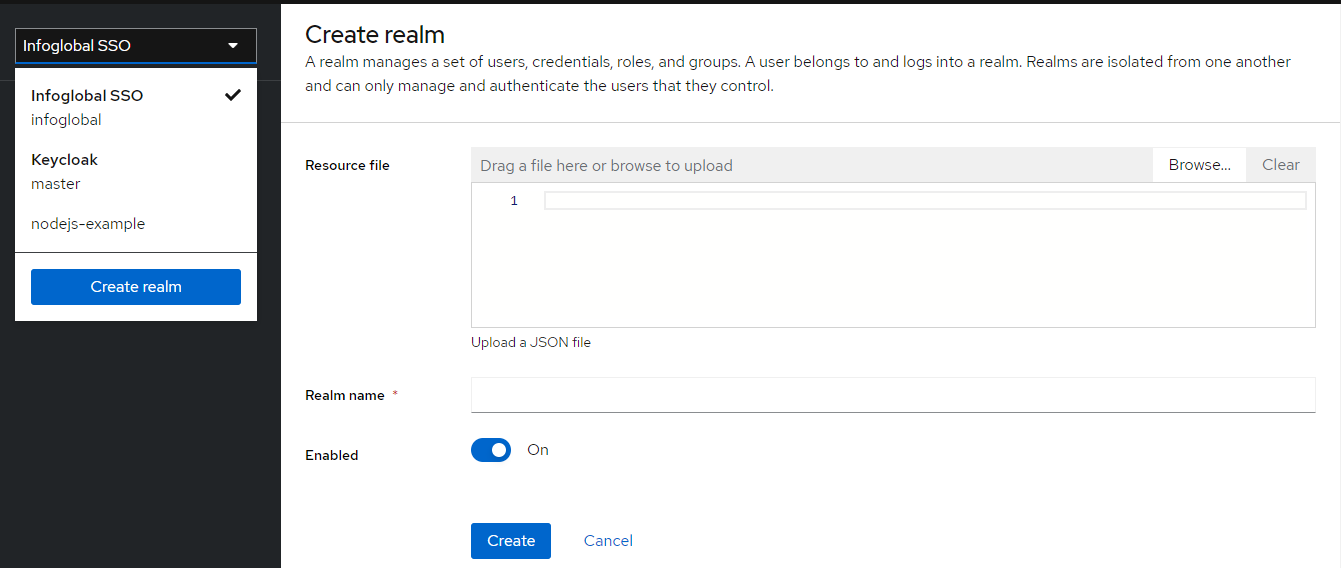
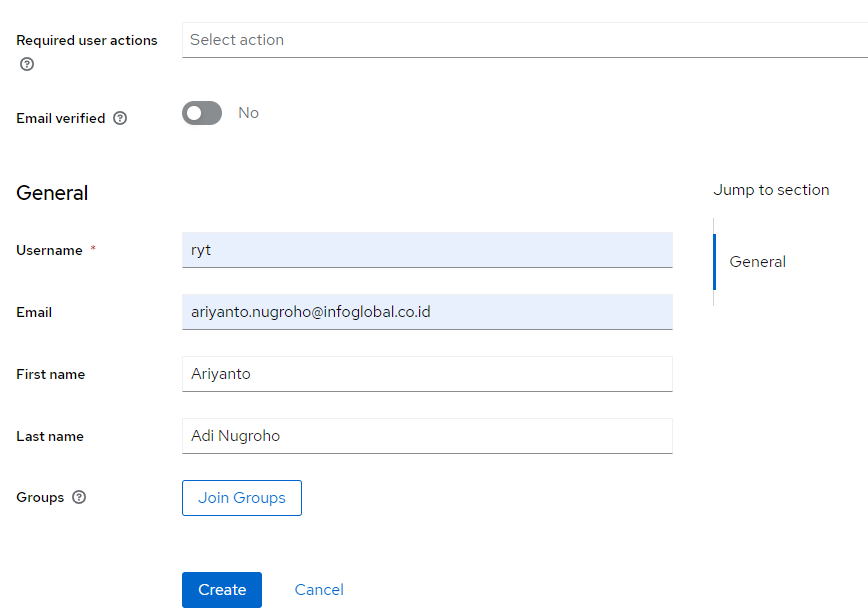
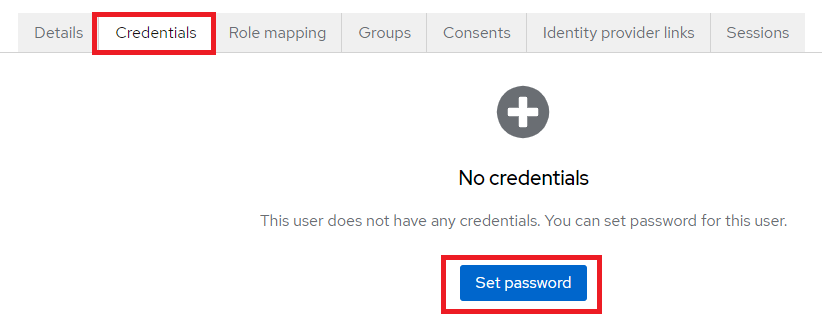
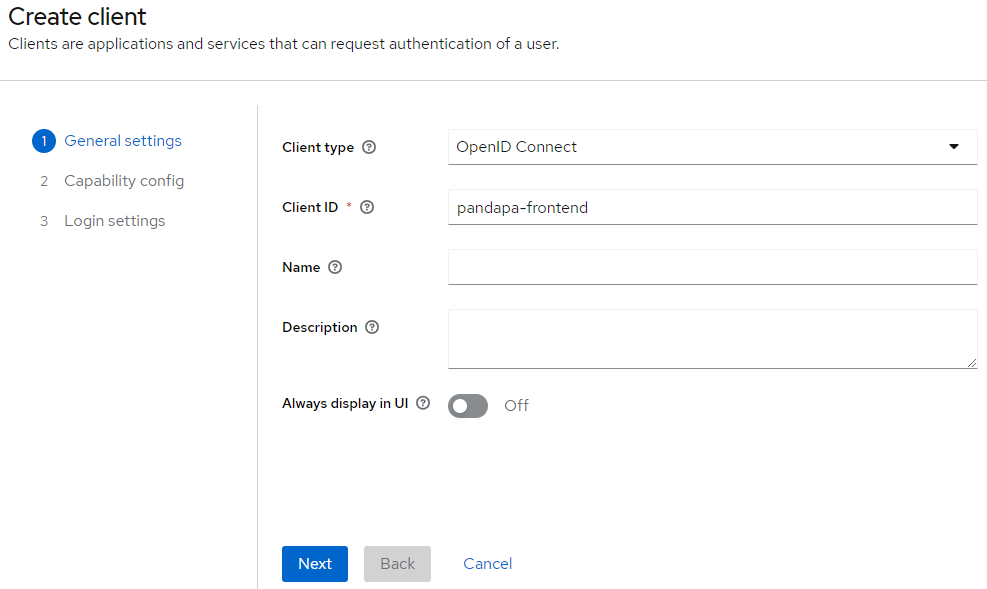
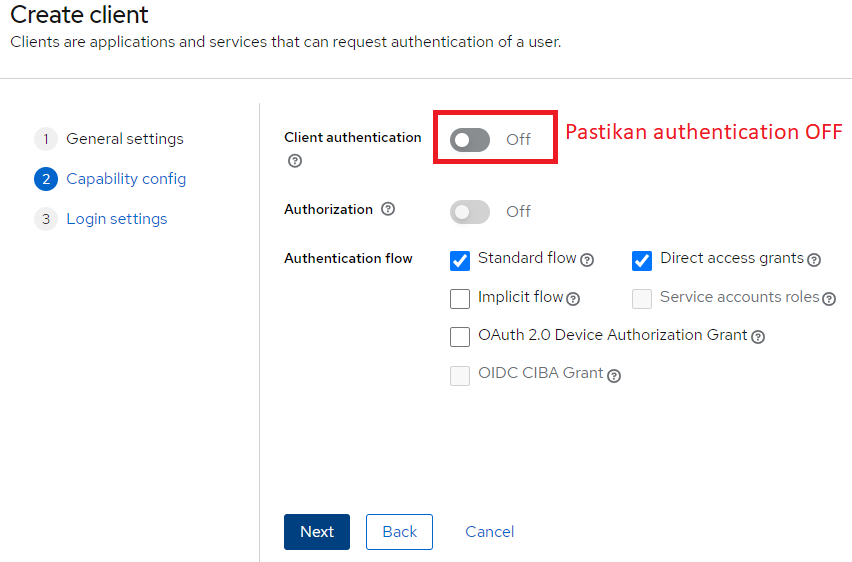
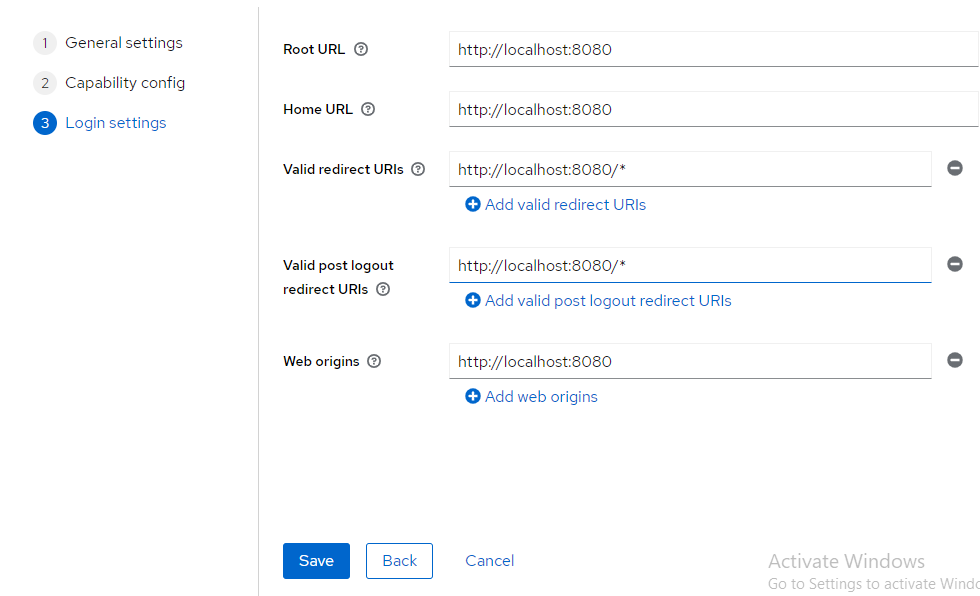
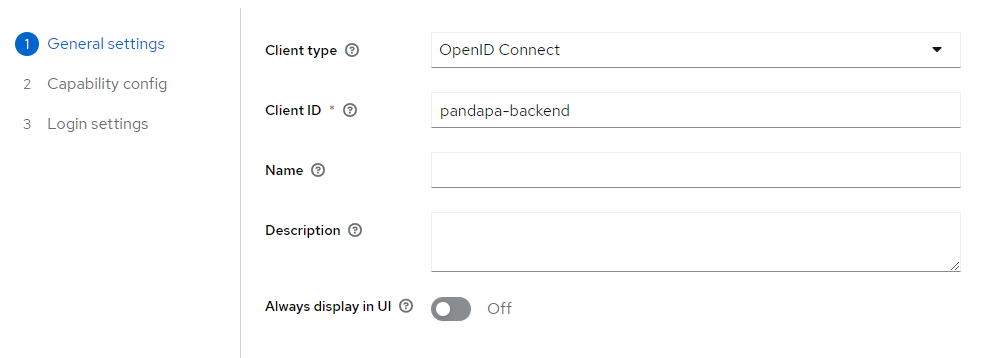
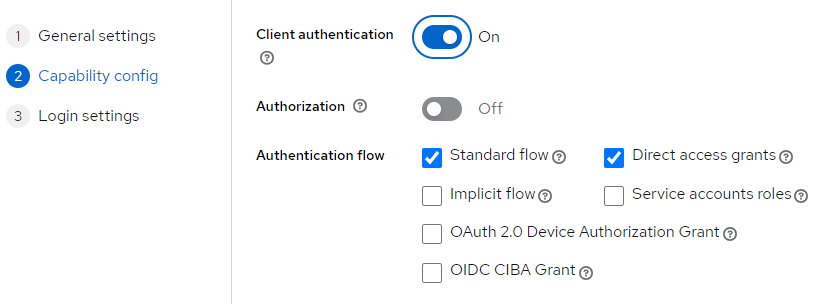
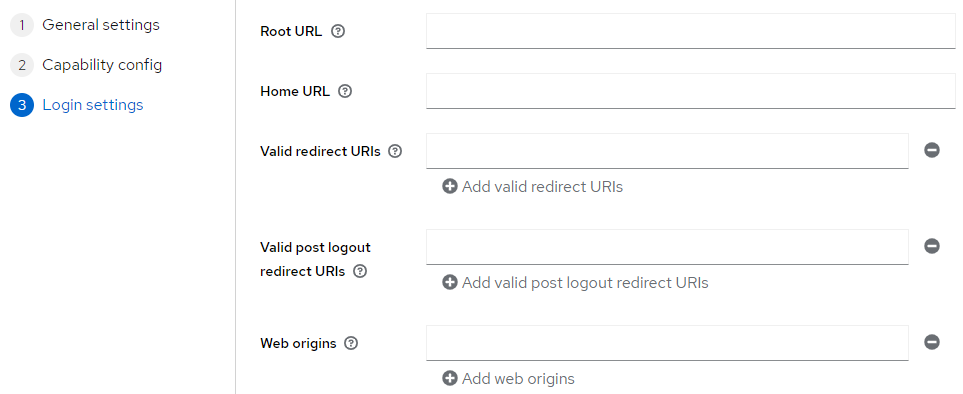
## Dokumentasi integrasi Pandapakita dengan Keycloak

Untuk melakukan integrasi dengan keycloak, pertama-tama setting dulu di keycloak  
1. Pilih / buat realm yang akan digunakan  


1. Buat user yang akan dipakai  
   
2. Setelah user berhasil dibuat, pindah ke tab “Credentials” untuk setting passwordnya.  
   
3. Setelah itu buat client untuk koneksi dengan Frontend Pandapakita

  
  
Pastikan authenticationnya off  


Setting url sesuai dengan url frontend  


1. Kemudian buat client untuk koneksi dengan Backend Pandapakita  
     
   Untuk client backend authentication bisa di aktifkan  
     
   Untuk setting url bisa dikosongkan saja  
   

Setelah setting keycloak selesai, mulai proses modifikasi untuk frontendnya.

Untuk Frontend yang harus dilakukan yaitu:  
1. install “keycloak-js”

|  |
| --- |
| npm install keycloak-js@25.0.1 |

1. Pertama-tama modif vuex store untuk menyimpan keterangan dari keycloak

|  |
| --- |
| state: {  authenticated: false,  token: '',  username: 'unidentified',  tokenValidity: ''  },  getters: {  getUserAuth: state => {  return {  authenticated: state.authenticated,  token: state.token,  username: state.username,  tokenValidity: state.tokenValidity  }  }  },  mutations: {  setUserAuth (state, payload) {  state.authenticated = payload.authenticated  state.token = payload.token  state.username = payload.username  state.tokenValidity = payload.tokenValidity  },  updateToken (state, payload) {  state.tokenValidity = payload.tokenValidity  state.token = payload.token  },  } |

Pada contoh saya membuat 4 variabel yaitu   
- “authenticated” untuk menyimpan status apakah login sudah terautentikasi  
- “token”untuk meyimpan token kiriman dari keycloak yang akan digunakan untuk memanggil backend  
- “username” untuk meyimpan username yang digunakan untuk login pada keycloak  
- “token validity” untuk meyimpan keterangan sisa waktu sebelum token expired

1. Buat file baru “KeycloakService.js” untuk koneksi dengan client keycloak

|  |
| --- |
| import Keycloak from 'keycloak-js';  let initOptions = {  url: 'https://sso.infoglobal.id:8443/',  realm: 'infoglobal',  clientId: 'pandapa-frontend'  }  const keycloakInstance = new Keycloak(initOptions); |

Import keycloak-js, kemudian definisikan koneksi keycloak dan panggil keycloak-js

|  |
| --- |
| const Init = (initCallback, refreshTokenCallback) => {  keycloakInstance.init({onLoad: 'check-sso', checkLoginIframe: false}).then((auth) => {  if (auth) {  initCallback({  authenticated: true,  token: keycloakInstance.token,  username: keycloakInstance.tokenParsed?.preferred\_username,  tokenParsed: keycloakInstance.tokenParsed,  tokenValidity: Math.round((keycloakInstance.tokenParsed.exp + keycloakInstance.timeSkew - new Date().getTime() / 1000)/60) + ' minutes'  })  } else {  initCallback({  authenticated: false,  token: '',  username: '',  tokenParsed: '',  tokenValidity: ''  })  }  setInterval(() =>{  if (keycloakInstance.authenticated) {  keycloakInstance.updateToken().then((refreshed)=>{  if (refreshed) {  refreshTokenCallback('Token refreshed'+ refreshed, keycloakInstance.token);  } else {  refreshTokenCallback('Token not refreshed, valid for '  + Math.round((keycloakInstance.tokenParsed.exp + keycloakInstance.timeSkew - new Date().getTime() / 1000)/60) + ' minutes',  keycloakInstance.token);  }  });  }  }, 60000)  });  } |

Disini saya membuat function untuk menginisialisasi keycloak berserta timer untuk merefresh tokennya

|  |
| --- |
| const Login = () => {  keycloakInstance.login()  }  const Logout = () => {  keycloakInstance.logout()  } |

Kemudian membuat function untuk login dan logoutnya

|  |
| --- |
| const KeyCloakService = {  CallInit: Init,  CallLogin: Login,  CallLogout: Logout  };  export default KeyCloakService; |

Export ketiga function diatas agar bisa dipanggil dari luar

1. Modif file main.js untuk memasukkan KeycloakService.js

|  |
| --- |
| import KeyCloakService from './security/KeycloakService'  KeyCloakService.CallInit(renderApp, refreshToken) |

Pada main panggil initialize yang sudah dibuat sebelumnya.  
Kemudian buatkan function render dan refreshtoken untuk menghandle callback dari KeycloakService

|  |
| --- |
| const renderApp = (userAuth) => {  new Vue({  router,  store: authStore,  render: h => h(App)  }).$mount('#app')  authStore.commit('setUserAuth', userAuth)  } |

|  |
| --- |
| const refreshToken = (tokenValidity, token) => {  const payload = {  tokenValidity: tokenValidity,  token: token  }  authStore.commit('updateToken', payload)  } |

1. Implementasikan button untuk login / logout lewat keycloak

|  |
| --- |
| <button @click="login">Login</button> <button @click="logout">Logout</button>  <script>  import KeyCloakService from './security/KeycloakService';  export default {  methods: {  login() {  KeyCloakService.CallLogin();  },  logout() {  KeyCloakService.CallLogout();  }  }  }  </script> |

1. Untuk mengakses backend tambahkan authorization pada headers saat memanggil api backend

|  |
| --- |
| const tokenAuth = 'Bearer ' + this.$store.state.token  axios.get('http://localhost:3000/personel/info', {  "headers": { "Authorization": tokenAuth }  }) |

Setelah modif frontend sekarang beralih modif di backendnya

1. Install keycloak-connect

|  |
| --- |
| npm install keycloak-connect@25.0.1 |

1. Buat file “keycloak.js” sebagai middleware dengan keycloak

|  |
| --- |
| const Keycloak = require("keycloak-connect");  const config = {  "realm": "infoglobal",  "auth-server-url": "https://sso.infoglobal.id:8443/",  "ssl-required": "external",  "resource": "pandapa-backend",  }  module.exports = new Keycloak({}, config); |

Import keycloak-connect, kemudian definisikan koneksi keycloak

1. Pada index.js atau router.js use.middleware yang sudah dibuat sebelumnya

|  |
| --- |
| const keycloak = require("#middlewares/keycloak");  app.use(keycloak.middleware()); |

1. Kemudian pada routernya ditambahkan keycloak.protect()

|  |
| --- |
| router.get("/…",  keycloak.protect()],  async ( req, res, next) => {…}); |