OpenAEV-Platform / **openaev** Public

Code | Issues 609 | Pull requests 66 | Discussions | Actions | Projects

Commit `3430fe2`

savacano28 authored on Jan 3, 2025 Verified

```
[Backend] Fix password reset for users who are not logged in (#1963)
```

`master` (#2141) · 2.3.3 ••• 1.11.0

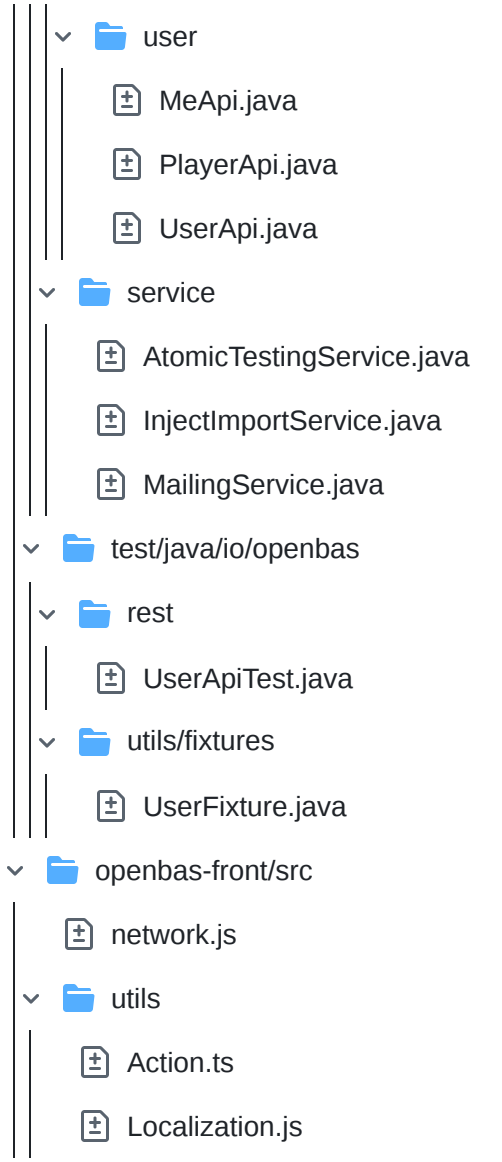
1 parent `f257e8e` commit `3430fe2`

18 files changed +150 -29 lines changed

Top

Filter files…

- openbas-api/src
  - main/java/io/openbas
    - rest
      - document
        - DocumentApi.java
      - exercise
        - ExerciseApi.java
      - helper
        - RestBehavior.java
      - inject
        - InjectApi.java
      - objective
        - ExerciseObjectiveApi.java
        - ScenarioObjectiveApi.java
      - team
        - TeamApi.java



18 files changed +150 -29 lines changed

Search within code

...a/io/openbas/rest/document/DocumentApi.java

```
@@ -349,7 +349,8 @@ public Document updateDocumentInformation(
                        !exercise.isUserHasAccess(
                            userRepository
                                .findById(currentUser().getId())
-                               .orElseThrow(ElementNotFoundException::new)))
+                               .orElseThrow(
+                                   () -> new ElementNotFoundException("Current user not found"))))
                .map(Exercise::getId);
        List<String> askExerciseIds =
```

```
355  356          Stream.concat(askExerciseIdsStream,
                  input.getExerciseIds().stream()).distinct().toList();
     @@ -370,7 +371,8 @@ public Document updateDocumentInformation(
370  371                      !scenario.isUserHasAccess(
371  372                          userRepository
372  373                              .findById(currentUser().getId())
373       -                              .orElseThrow(ElementNotFoundException::new)))
          374 +                              .orElseThrow(
          375 +                                  () -> new ElementNotFoundException("Current
                  user not found"))))
374  376                  .map(Scenario::getId);
375  377      List<String> askScenarioIds =
376  378          Stream.concat(askScenarioIdsStream,
                  input.getScenarioIds().stream()).distinct().toList();
```

...a/io/openbas/rest/exercise/ExerciseApi.java

```
     @@ -127,7 +127,9 @@ public Log createLog(@PathVariable String exerciseId,
                  @Valid @RequestBody LogCre
127  127          log.setExercise(exercise);
128  128          log.setTags(iterableToSet(tagRepository.findAllById(input.getTagIds())));
129  129          log.setUser(
130       -
                  userRepository.findById(currentUser().getId()).orElseThrow(ElementNotFoundExcep
                  tion::new));
          130 +          userRepository
          131 +              .findById(currentUser().getId())
          132 +              .orElseThrow(() -> new ElementNotFoundException("Current user not
                  found")));
131  133          return exerciseLogRepository.save(log);
132  134        }
133  135
     @@ -172,7 +174,7 @@ public Dryrun createDryrun(
172  174                  ? List.of(
173  175                      userRepository
174  176                          .findById(currentUser().getId())
175       -                          .orElseThrow(ElementNotFoundException::new))
          177 +                          .orElseThrow(() -> new ElementNotFoundException("Current
                  user not found")))
```

```
176  178                 : fromIterable(userRepository.findAllById(userIds));
177  179           return dryrunService.provisionDryrun(exercise, users, input.getName());
178  180         }
```

...va/io/openbas/rest/helper/RestBehavior.java

```
@@ -163,21 +163,29 @@ public ResponseEntity<ErrorMessage> handleAlreadyExistingException(AlreadyExisti
163  163
164  164       // --- Open channel access
165  165       public User impersonateUser(UserRepository userRepository, Optional<String> userId) {
166       -     if (currentUser().getId().equals(ANONYMOUS)) {
167       -       if (userId.isPresent()) {
168       -         return userRepository.findById(userId.get()).orElseThrow();
     166  +     if (ANONYMOUS.equals(currentUser().getId())) {
     167  +       if (userId.isEmpty()) {
     168  +         throw new UnsupportedOperationException(
     169  +             "User must be logged or dynamic player is required");
169  170         }
170       -       throw new UnsupportedOperationException("User must be logged or dynamic player is required");
     171  +       return userRepository
     172  +           .findById(userId.get())
     173  +           .orElseThrow(() -> new ElementNotFoundException("User not found"));
171  174       }
172       -     return userRepository.findById(currentUser().getId()).orElseThrow();
     175  +     return userRepository
     176  +         .findById(currentUser().getId())
     177  +         .orElseThrow(() -> new ElementNotFoundException("Current user not found"));
173  178       }
174  179
175  180       public void checkUserAccess(UserRepository userRepository, String userId) {
176  181         User askedUser = userRepository.findById(userId).orElseThrow();
177  182         if (askedUser.getOrganization() != null) {
178  183           OpenBASPrincipal currentUser = currentUser();
179  184           if (!currentUser.isAdmin()) {
180       -           User local = userRepository.findById(currentUser.getId()).orElseThrow();
```

```
      185 +         User local =
      186 +             userRepository
      187 +                 .findById(currentUser.getId())
      188 +                 .orElseThrow(() -> new ElementNotFoundException("Current user
          not found"));
181   189           List<String> localOrganizationIds =
182   190               local.getGroups().stream()
183   191                   .flatMap(group -> group.getOrganizations().stream())
          @@ -194,7 +202,10 @@ public void checkOrganizationAccess(UserRepository
          userRepository, String organi
194   202       if (organizationId != null) {
195   203         OpenBASPrincipal currentUser = currentUser();
196   204         if (!currentUser.isAdmin()) {
197       -         User local =
          userRepository.findById(currentUser.getId()).orElseThrow();
      205 +         User local =
      206 +             userRepository
      207 +                 .findById(currentUser.getId())
      208 +                 .orElseThrow(() -> new ElementNotFoundException("Current user
          not found"));
198   209           List<String> localOrganizationIds =
199   210               local.getGroups().stream()
200   211                   .flatMap(group -> group.getOrganizations().stream())
```

.../java/io/openbas/rest/inject/InjectApi.java

```
          @@ -319,7 +319,9 @@ public Inject createInjectForExercise(
319   319       // Get common attributes
320   320       Inject inject = input.toInject(injectorContract);
321   321       inject.setUser(
322       -
          userRepository.findById(currentUser().getId()).orElseThrow(ElementNotFoundExcep
          tion::new));
      322 +         userRepository
      323 +             .findById(currentUser().getId())
      324 +             .orElseThrow(() -> new ElementNotFoundException("Current user not
          found")));
323   325       inject.setExercise(exercise);
324   326       // Set dependencies
325   327       if (input.getDependsOn() != null) {
```

```
@@ -489,7 +491,8 @@ public List<Inject> nextInjectsToExecute(@RequestParam Optional<Integer> size) {
489 491                         .isUserHasAccess(
490 492                             userRepository
491 493                                 .findById(currentUser().getId())
492     -                                 .orElseThrow(ElementNotFoundException::new)))
    494 +                                 .orElseThrow(
    495 +                                     () -> new ElementNotFoundException("Current user not found"))))
493 496             // Order by near execution
494 497             .sorted(Inject.executionComparator)
495 498             // Keep only the expected size
@@ -552,7 +555,7 @@ public Inject createInjectForScenario(
552 555         inject.setUser(
553 556             this.userRepository
554 557                 .findById(currentUser().getId())
555     -                 .orElseThrow(ElementNotFoundException::new));
    558 +                 .orElseThrow(() -> new ElementNotFoundException("Current user not found")));
556 559         inject.setScenario(scenario);
557 560         // Set dependencies
558 561         if (input.getDependsOn() != null) {
```

...as/rest/objective/ExerciseObjectiveApi.java

```
@@ -101,7 +101,9 @@ public Evaluation createEvaluation(
101 101         Objective objective = resolveRelation(objectiveId, objectiveRepository);
102 102         evaluation.setObjective(objective);
103 103         evaluation.setUser(
104     -             userRepository.findById(currentUser().getId()).orElseThrow(ElementNotFoundException::new));
    104 +             userRepository
    105 +                 .findById(currentUser().getId())
    106 +                 .orElseThrow(() -> new ElementNotFoundException("Current user not found")));
105 107         Evaluation result = evaluationRepository.save(evaluation);
106 108         objective.setUpdatedAt(now());
107 109         objectiveRepository.save(objective);
```

...as/rest/objective/ScenarioObjectiveApi.java

```
@@ -101,7 +101,9 @@ public Evaluation createEvaluation(
101 101         Objective objective = resolveRelation(objectiveId, objectiveRepository);
102 102         evaluation.setObjective(objective);
103 103         evaluation.setUser(
104     -         userRepository.findByIdOrThrow(currentUser().getId(), "User not found"));
    104 +         userRepository
    105 +             .findById(currentUser().getId())
    106 +             .orElseThrow(() -> new ElementNotFoundException("Current user not found")));
105 107         Evaluation result = evaluationRepository.save(evaluation);
106 108         objective.setUpdatedAt(now());
107 109         objectiveRepository.save(objective);
```

...main/java/io/openbas/rest/team/TeamApi.java

```
@@ -69,7 +69,9 @@ public Iterable<TeamSimple> getTeams() {
69 69         } else {
70 70           // We get the teams that are linked to the organizations we are part of
71 71           User local =
72     -           userRepository.findByIdOrThrow(currentUser.getId(), "User not found");
    72 +           userRepository
    73 +               .findById(currentUser.getId())
    74 +               .orElseThrow(() -> new ElementNotFoundException("Current user not found"));
73 75           List<String> organizationIds =
74 76               local.getGroups().stream()
75 77                   .flatMap(group -> group.getOrganizations().stream())
```

...c/main/java/io/openbas/rest/user/MeApi.java

```
@@ -71,14 +71,16 @@ public ResponseEntity<Object> logout() {
71 71       public User me() {
```

```
72  72          return userRepository
73  73              .findById(currentUser().getId())
74      -         .orElseThrow(ElementNotFoundException::new);
    74  +         .orElseThrow(() -> new ElementNotFoundException("Current user not found"));
75  75        }
76  76
77  77        @Secured(ROLE_USER)
78  78        @PutMapping("/api/me/profile")
79  79        public User updateProfile(@Valid @RequestBody UpdateProfileInput input) {
80  80          User user =
81      -         userRepository.findById(currentUser().getId()).orElseThrow(ElementNotFoundException::new);
    81  +         userRepository
    82  +             .findById(currentUser().getId())
    83  +             .orElseThrow(() -> new ElementNotFoundException("Current user not found"));
82  84          user.setUpdateAttributes(input);
83  85          user.setOrganization(
84  86              updateRelation(input.getOrganizationId(), user.getOrganization(), organizationRepository));
        @@ -91,7 +93,9 @@ public User updateProfile(@Valid @RequestBody UpdateProfileInput input) {
91  93        @PutMapping("/api/me/information")
92  94        public User updateInformation(@Valid @RequestBody UpdateUserInfoInput input) {
93  95          User user =
94      -         userRepository.findById(currentUser().getId()).orElseThrow(ElementNotFoundException::new);
    96  +         userRepository
    97  +             .findById(currentUser().getId())
    98  +             .orElseThrow(() -> new ElementNotFoundException("Current user not found"));
95  99          user.setUpdateAttributes(input);
96 100          User savedUser = userRepository.save(user);
97 101          sessionManager.refreshUserSessions(savedUser);
        @@ -103,7 +107,9 @@ public User updateInformation(@Valid @RequestBody UpdateUserInfoInput input) {
```

```
103  107      public User updatePassword(@Valid @RequestBody UpdateMePasswordInput input)
104  108          throws InputValidationException {
105  109        User user =
106       -
                 userRepository.findById(currentUser().getId()).orElseThrow(ElementNotFoundExcep
                 tion::new);
          110  +         userRepository
          111  +             .findById(currentUser().getId())
          112  +             .orElseThrow(() -> new ElementNotFoundException("Current user not
                 found"));
107  113        if (userService.isUserPasswordValid(user, input.getCurrentPassword())) {
108  114          user.setPassword(userService.encodeUserPassword(input.getPassword()));
109  115          return userRepository.save(user);
                 @@ -118,7 +124,9 @@ public User updatePassword(@Valid @RequestBody
                 UpdateMePasswordInput input)
118  124      public Token renewToken(@Valid @RequestBody RenewTokenInput input)
119  125          throws InputValidationException {
120  126        User user =
121       -
                 userRepository.findById(currentUser().getId()).orElseThrow(ElementNotFoundExcep
                 tion::new);
          127  +         userRepository
          128  +             .findById(currentUser().getId())
          129  +             .orElseThrow(() -> new ElementNotFoundException("Current user not
                 found"));
122  130        Token token =
123  131
                 tokenRepository.findById(input.getTokenId()).orElseThrow(ElementNotFoundExcepti
                 on::new);
124  132        if (!user.equals(token.getUser())) {
```

...in/java/io/openbas/rest/user/PlayerApi.java

```
                 @@ -57,7 +57,9 @@ public Iterable<RawPlayer> players() {
57   57           players = fromIterable(userRepository.rawAllPlayers());
58   58         } else {
59   59           User local =
60        -
                 userRepository.findById(currentUser.getId()).orElseThrow(ElementNotFoundExceptio
                 n::new);
```

```
      60  +           userRepository
      61  +               .findById(currentUser.getId())
      62  +               .orElseThrow(() -> new ElementNotFoundException("Current user not
          found"));
61    63            List<String> organizationIds =
62    64                local.getGroups().stream()
63    65                    .flatMap(group -> group.getOrganizations().stream())
```

...main/java/io/openbas/rest/user/UserApi.java

```
          @@ -33,6 +33,7 @@
33    33    import org.springframework.data.domain.Page;
34    34    import org.springframework.data.domain.Pageable;
35    35    import org.springframework.data.jpa.domain.Specification;
      36  + import org.springframework.http.ResponseEntity;
36    37    import org.springframework.security.access.AccessDeniedException;
37    38    import org.springframework.security.access.annotation.Secured;
38    39    import org.springframework.security.authentication.BadCredentialsException;
          @@ -87,7 +88,7 @@ public User login(@Valid @RequestBody LoginUserInput
          input) {
87    88      }
88    89
89    90      @PostMapping("/api/reset")
90        -   public void passwordReset(@Valid @RequestBody ResetUserInput input) {
      91  +   public ResponseEntity<?> passwordReset(@Valid @RequestBody ResetUserInput
          input) {
91    92        Optional<User> optionalUser =
          userRepository.findByEmailIgnoreCase(input.getLogin());
92    93        if (optionalUser.isPresent()) {
93    94          User user = optionalUser.get();
          @@ -116,7 +117,9 @@ public void passwordReset(@Valid @RequestBody
          ResetUserInput input) {
116   117         }
117   118         // Store in memory reset token
118   119         resetTokenMap.put(resetToken, user.getId());
      120 +       return ResponseEntity.ok().build();
119   121       }
      122 +     return ResponseEntity.badRequest().build();
120   123     }
121   124
```

```
122  125      @PostMapping("/api/reset/{token}")
```

...o/openbas/service/AtomicTestingService.java

```
@@ -103,7 +103,10 @@ public InjectResultOverviewOutput createOrUpdate(AtomicTestingInput input, Strin
103  103      injectToSave.setAllTeams(input.isAllTeams());
104  104      injectToSave.setDescription(input.getDescription());
105  105      injectToSave.setDependsDuration(0L);
106      -    injectToSave.setUser(userRepository.findById(currentUser().getId()).orElseThrow());
     106  +    injectToSave.setUser(
     107  +        userRepository
     108  +            .findById(currentUser().getId())
     109  +            .orElseThrow(() -> new ElementNotFoundException("Current user not found")));
107  110      injectToSave.setExercise(null);
108  111
109  112      // Set dependencies
```

...io/openbas/service/InjectImportService.java

```
@@ -710,7 +710,10 @@ private ImportRow importRow(
710  710                              mapPatternByAllTeams));
711  711              });
712  712      // The user is the one doing the import
713      -    inject.setUser(userRepository.findById(currentUser().getId()).orElseThrow());
     713  +    inject.setUser(
     714  +        userRepository
     715  +            .findById(currentUser().getId())
     716  +            .orElseThrow(() -> new ElementNotFoundException("Current user not found")));
714  717      // No exercise yet
715  718      inject.setExercise(null);
716  719      // No dependencies
```

...java/io/openbas/service/MailingService.java

```
@@ -1,5 +1,6 @@
package io.openbas.service;

+ import static io.openbas.config.OpenBASAnonymous.ANONYMOUS;
import static io.openbas.config.SessionHelper.currentUser;

import com.fasterxml.jackson.databind.ObjectMapper;
@@ -74,7 +75,15 @@ public void sendEmail(
          .ifPresent(
              injectorContract -> {
                inject.setContent(this.mapper.valueToTree(emailContent));
-               inject.setUser(this.userRepository.findById(currentUser().getId()).orElseThrow()
-               );
+
+               // When resetting the password, the user is not logged in (anonymous),
+               // so there's no need to add the user to the inject.
+               if (!ANONYMOUS.equals(currentUser().getId())) {
+                 inject.setUser(
+                     this.userRepository
+                         .findById(currentUser().getId())
+                         .orElseThrow(() -> new ElementNotFoundException("Current user not found")));
+               }

                exercise.ifPresent(inject::setExercise);
```

.../test/java/io/openbas/rest/UserApiTest.java

```
@@ -2,7 +2,12 @@

 import static io.openbas.utils.JsonUtils.asJsonString;
 import static io.openbas.utils.fixtures.UserFixture.EMAIL;
+ import static org.junit.jupiter.api.Assertions.assertEquals;
```

```
5   6    import static org.junit.jupiter.api.TestInstance.Lifecycle.PER_CLASS;
    7  + import static org.mockito.ArgumentMatchers.any;
    8  + import static org.mockito.ArgumentMatchers.anyString;
    9  + import static org.mockito.Mockito.never;
    10 + import static org.mockito.Mockito.verify;
6   11   import static
         org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
7   12   import static
         org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;
8   13   import static
         org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
        @@ -11,10 +16,15 @@
11  16   import io.openbas.database.model.User;
12  17   import io.openbas.database.repository.UserRepository;
13  18   import io.openbas.rest.user.form.login.LoginUserInput;
    19 + import io.openbas.rest.user.form.login.ResetUserInput;
14  20   import io.openbas.rest.user.form.user.CreateUserInput;
    21 + import io.openbas.service.MailingService;
15  22   import io.openbas.utils.fixtures.UserFixture;
    23 + import java.util.List;
16  24   import org.junit.jupiter.api.*;
    25 + import org.mockito.ArgumentCaptor;
17  26   import org.springframework.beans.factory.annotation.Autowired;
    27 + import org.springframework.boot.test.mock.mockito.MockBean;
18  28   import org.springframework.http.MediaType;
19  29   import org.springframework.security.test.context.support.WithMockUser;
20  30   import org.springframework.test.web.servlet.MockMvc;
        @@ -28,6 +38,8 @@ class UserApiTest extends IntegrationTest {
28  38
29  39     @Autowired private UserRepository userRepository;
30  40
    41 +   @MockBean private MailingService mailingService;
    42 +
31  43     @BeforeAll
32  44     public void setup() {
33  45       // Create user
        @@ -143,4 +155,45 @@ void
        given_known_create_user_in_uppercase_input_should_return_conflict() throws
143 155              .andExpect(status().isConflict());
144 156       }
```

```
145  157       }
     158  +
     159  +   @Nested
     160  +   @DisplayName("Reset Password from I forget my pwd option")
     161  +   class ResetPassword {
     162  +     @DisplayName("With a known email")
     163  +     @Test
     164  +     void resetPassword() throws Exception {
     165  +       // -- PREPARE --
     166  +       ResetUserInput input = UserFixture.getResetUserInput();
     167  +
     168  +       // -- EXECUTE --
     169  +       mvc.perform(
     170  +               post("/api/reset")
     171  +                   .contentType(MediaType.APPLICATION_JSON)
     172  +                   .content(asJsonString(input)))
     173  +           .andExpect(status().isOk());
     174  +
     175  +       // -- ASSERT --
     176  +       ArgumentCaptor<List<User>> userCaptor =
              ArgumentCaptor.forClass(List.class);
     177  +       verify(mailingService).sendEmail(anyString(), anyString(),
              userCaptor.capture());
     178  +       assertEquals(EMAIL, userCaptor.getValue().get(0).getEmail());
     179  +     }
     180  +
     181  +     @DisplayName("With a unknown email")
     182  +     @Test
     183  +     void resetPasswordWithUnknownEmail() throws Exception {
     184  +       // -- PREPARE --
     185  +       ResetUserInput input = UserFixture.getResetUserInput();
     186  +       input.setLogin("unknown@filigran.io");
     187  +
     188  +       // -- EXECUTE --
     189  +       mvc.perform(
     190  +               post("/api/reset")
     191  +                   .contentType(MediaType.APPLICATION_JSON)
     192  +                   .content(asJsonString(input)))
     193  +           .andExpect(status().isBadRequest());
     194  +
```

```
      195 +       // -- ASSERT --
      196 +       verify(mailingService, never()).sendEmail(anyString(), anyString(),
          +   any(List.class));
      197 +     }
      198 +   }
146   199     }
```

.../io/openbas/utils/fixtures/UserFixture.java

```
          @@ -2,6 +2,7 @@
2     2
3     3     import io.openbas.database.model.User;
4     4     import io.openbas.rest.user.form.login.LoginUserInput;
      5   + import io.openbas.rest.user.form.login.ResetUserInput;
5     6     import org.springframework.security.crypto.argon2.Argon2PasswordEncoder;
6     7
7     8     public class UserFixture {
          @@ -45,4 +46,11 @@ public static User getSavedUser() {
45    46         user.setId("saved-user-id");
46    47         return user;
47    48       }
      49   +
      50   +   public static ResetUserInput getResetUserInput() {
      51   +     ResetUserInput resetUserInput = new ResetUserInput();
      52   +     resetUserInput.setLogin(EMAIL);
      53   +
      54   +     return resetUserInput;
      55   +   }
48    56     }
```

openbas-front/src/network.js

```
          @@ -9,8 +9,10 @@ export const api = (schema) => {
9     9       // Intercept to apply schema and test unauthorized users
10    10      instance.interceptors.response.use(
11    11        (response) => {
12        -       if (schema) {
13        -         response.data = normalize(response.data, schema);
      12  +       if (response.data && schema) {
      13  +         if (typeof response.data === 'object') {
```

```
      14  +           response.data = normalize(response.data, schema);
      15  +         }
14    16            }
15    17            return response;
16    18          },
```

openbas-front/src/utils/Action.ts

```
                  @@ -50,6 +50,8 @@ const notifyError = (error: AxiosError) => {
50    50            // Do not notify the user, as a 401 error will already trigger a disconnection, as 404 already handle inside the app
51    51          } else if (error.status === 409) {
52    52            MESSAGING$.notifyError(intl.formatMessage({ id: 'The element already exists' }));
      53  +       } else if (error.status === 400) {
      54  +         MESSAGING$.notifyError(intl.formatMessage({ id: 'Bad request' }));
53    55          } else if (error.status === 500) {
54    56            MESSAGING$.notifyError(intl.formatMessage({ id: 'Internal error' }));
55    57          } else if (error.message) {
                  @@ -166,6 +168,7 @@ export const postReferential = (schema: Schema | null, uri: string, data: unknow
166   168           .post(buildUri(uri), data)
167   169           .then((response) => {
168   170             dispatch({ type: Constants.DATA_FETCH_SUCCESS, payload: response.data });
      171 +         notifySuccess('The element has been successfully updated');
169   172             return response.data;
170   173           })
171   174           .catch((error) => {
```

openbas-front/src/utils/Localization.js

```
                  @@ -1425,6 +1425,7 @@ const i18n = {
1425  1425            'Childrens': 'Enfants',
1426  1426            'Interactive view': 'Vue interactive',
1427  1427            'Execution successful': 'Exécution réussie',
      1428  +         'Bad request': 'Mauvaise requête',
1428  1429            'Internal error': 'Erreur interne',
1429  1430            'The element has been successfully updated': 'L\'élément a été mis à jour avec succès',
```

```
1430  1431          'The element has been successfully deleted': 'L\'élément a été supprimé
                    avec succès',
                    @@ -2812,6 +2813,7 @@ const i18n = {
2812  2813          'Execution successful': '执行成功',
2813  2814          'The element has been successfully updated': '元素已成功更新',
2814  2815          'The element has been successfully deleted': '元素已成功删除',
      2816  +       'Bad request': '错误的请求',
2815  2817          'Internal error': '内部错误 ',
2816  2818          'No data to display': '没有可显示的数据',
2817  2819          'No simulation in this platform yet': '此平台尚未提供模拟功能',
                    @@ -2987,6 +2989,7 @@ const i18n = {
2987  2989          'Do you want to launch this atomic testing: {title}?': 'Do you want to
                    launch this atomic testing: {title}?',
2988  2990          'Do you want to relaunch this atomic testing: {title}?': 'Do you want
                    to relaunch this atomic testing: {title}?',
2989  2991          'The element already exists': 'The element already exists',
      2992  +       'Bad request': 'Bad request',
2990  2993          'Internal error': 'Internal error',
2991  2994          'Something went wrong. Please refresh the page or try again later.':
                    'Something went wrong. Please refresh the page or try again later.',
2992  2995        },
```

Comments 0

Please sign in to comment.