

Peer Code Review

Date	:	28/03/2022
Version	:	0.2
State	:	
Author	:	Nikola Stankov

Version history

Version	Date	Author(s)	Changes	State
0.1	15/06/2022	Nikola Stankov	Initial version	Finish

Distribution

Version	Date	Receivers

Table of contents

1. Introduction	4
2. Peers code reviews	4
3. General code review	9
3.1. Code testability	9
3.2. Architecture	11
4. Conclusion	13

1. Introduction

This document is meant to showcase my feedback on my team members for the group project we are building for the company Ivanti. The document serves as proof that I have reviewed my peers' code and made suggestions or gave general feedback about their work.

2. Peers code reviews

est version ▾ 16 files +411 -8 Expand all files ⚙ ▾

FrontEnd/ivanti-marketplace/src/Components/other/PopupContent.js 0 → 100644 +62 -0 Viewed ⋮

```
1 + import React from "react";
2 + import PopupNavigation from "../PopupNavigation";
3 + import {useState, useEffect} from "react";
4 + import PopupPages from "../PopupPages";
5 +
6 + const PopupContent = ({ close }) => {
7 +   let totalSteps = 6;
8 +   const [step, setStep] = useState(1);
9 +   const [buttons, setButtons] = useState(['next']);
10 +
11 +   useEffect(() => {
```

Stankov,Nikola N.M. @I453582 · 6 minutes ago Maintainer ✓ 😊 💬 ✎ ⋮

Logic for rendering the buttons seems pretty well-thought. Nice.

Edited by Stankov,Nikola N.M. 5 minutes ago

👍 1 😊

Reply... Resolve thread 📎

Assignee Bandell,Borek B. (Busy) @I489843

0 Reviewers None

Milestone None

Time tracking No estimate or time spent

Labels None

Lock merge request Unlocked

1 participant

Notifications


Code review on my colleague Borek.

latest version ▾

16 files +411 -8 Expand all files ⚙ ▾

FrontEnd/ivanti-marketplace/src/Components/other/PopupContent.js 0 → 100644 +62 -0 Viewed ⋮

```
38 + const handleKey = (event) => {
39 +   if (event.key === 'ArrowRight') {
40 +     onNext();
41 +   }
42 +   if (event.key === 'ArrowLeft') {
43 +     onBack();
44 +   }
45 + }
46 +
47 + return (
48 +   <div onKeyDown={handleKey} tabIndex="0" className="modal">
49 +     <a className="close" onClick={close}>
50 +       &times;
51 +     </a>
52 +     <div className="header"> Welcome! </div>
53 +     <div className="content">
54 +       { " " }
55 +     <PopupPages step={step} />
56 +     </div>
57 +     <PopupNavigation step={step} totalSteps={totalSteps} onBack=
      {onBack} onNext={onNext} buttons={buttons} />
58 +   </div>
59 + );
```


 Stankov,Nikola N.M. @I453582 · 4 minutes ago

Maintainer

👍 😊 💬 ✎ ⋮

Passing the state variables trough properties to the components. Good job, looks good!

Assignee

 Bandell,Borek B. (Busy) @I489843

Edit

0 Reviewers

None

Edit

Milestone

None

Edit

Time tracking

No estimate or time spent

?

Labels

None


Edit

Lock merge request

🔒 Unlocked

Edit

1 participant



Notifications

✕

Reference: I453582/group2project...


Code review on my colleague Borek.

test version ▾

16 files +411 -8 Expand all files ⚙ ▾

FrontEnd/ivanti-marketplace/src/Components/other/PopupNavigation.js 0 → 100644 +50 -0 Viewed ⋮

```
36 +   setNextButton(hiddenNextButton);
37 +   };
38 +
39 +   }, [props.buttons]);
40 +
41 +   return (
42 +     <div className="popup-navigation">
43 +       {backButton}
44 +       <p>{props.step} out of {props.totalSteps}</p>
45 +       {nextButton}
46 +     </div>
47 +   )
48 + }
49 +
50 + export default PopupNavigation;
```

 Stankov,Nikola N.M. @I453582 · 3 minutes ago

Maintainer

👍 😊 💬 ✎ ⋮


Not too much going on here looks fine.

Reply...

Resolve thread

🔗

Assignee

 Bandell,Borek B. (Busy) @I489843

Edit

0 Reviewers

None

Edit

Milestone

None

Edit

Time tracking

No estimate or time spent

?

Labels

None


Edit

Lock merge request

🔒 Unlocked

Edit

1 participant



Notifications

✕

Code review on my colleague Borek.

test version

6 files +117 -2

Expand all files

BackEnd/marketplace-backend/src/main/java/nl/fo
ntys/marketplacebackend/controller/PackageController.jav

+10 -0

Viewed

a

22

24

25

23

24

25

26

27

28

private final PackageService packageService;

private final TopRatedAlgorithm topRatedService;

// Get all packages

// GET /packages

Show 20 lines

Show all unchanged lines

Show 20 lines

76

77

78

79

80

81

82

83

84

@@ -76,4 +79,11 @@ public class PackageController {

return ResponseEntity.internalServerError().build();

}

@GetMapping("/topRated")

public ResponseEntity<List<PackageDto>> getTopRated() {

Stankov,Nikola N.M. @I453582 · 10 minutes ago

Maintainer

Using DTOs in the controller. Looks good.

1

0 Assignees

None - assign yourself

Edit

0 Reviewers

None

Edit

Milestone

None

Edit

Time tracking

No estimate or time spent

Labels

None

Edit

Lock merge request

Unlocked

Edit

1 participant

Notifications

Code review on my colleague Roel.

test version

6 files +117 -2

Expand all files

BackEnd/marketplace-backend/src/main/java/nl/fo
ntys/marketplacebackend/service/impl/TopRatedAlgorith
mImpl.java

+48 -0

Viewed

0 → 100644

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

+ @Primary

+ @RequiredArgsConstructor

+ public class TopRatedAlgorithmImpl implements TopRatedAlgorithm {

+ private final PackageServiceImpl packageService;

+

+ @Override

+ public List<PackageDto> getTopRated(){

+ List<PackageDto> topRatedPackages = new LinkedList<>();

+

+ List<PackageDto> packages = packageService.getAllPackages();

+ PackageDto[] packageArray = new PackageDto[packages.size()];

+ packages.toArray(packageArray);

+

+ Arrays.sort(packageArray, new SortByRatingAmount());

+

+ for(PackageDto packageDto : packageArray){

Stankov,Nikola N.M. @I453582 · 6 minutes ago

Maintainer

Maybe instead of for loop you could just use a .map to filter through the packages with rating >= 3.
Would be a little bit easier I think.

Reply...

Resolve thread

0 Assignees

None - assign yourself

Edit

0 Reviewers

None

Edit

Milestone

None

Edit

Time tracking

No estimate or time spent

Labels

None

Edit

Lock merge request

Unlocked

Edit

1 participant

Notifications

Reference: I453582/group2project...

Code review on my colleague Roel.

test version ▾

6 files +117 -2

Expand all files

⚙ ▾

FrontEnd/ivanti-marketplace/src/pages/Home.js +26 -2 Viewed

41 43 axios(config)

42 44 .then(function (response) {

⌵ Show 20 lines ⬆ Show all unchanged lines ⬆ Show 20 lines

54 56 @@ -54,6 +56,22 @@ const Home = (props) => {

55 57 }

56 58

59 + function getTopRatedPackages() {

60 + var config = {

61 + method: 'get',

62 + url: 'http://localhost:8080/packages/topRated',

63 + headers: { }

Stankov,Nikola N.M. @I453582 · 1 minute ago

Maintainer

✓ 😊 💬 ✎ ⋮

You could skip this line.

Reply...

Resolve thread

🔗

0 Assignees

None - assign yourself

Edit

0 Reviewers

None

Edit

Milestone

None

Edit

Time tracking

No estimate or time spent

?

Labels

None

Edit

Lock merge request

Unlocked

Edit

1 participant

Code review on my colleague Roel.

test version ▾

6 files +117 -2

Expand all files

⚙ ▾

FrontEnd/ivanti-marketplace/src/pages/Home.js +26 -2 Viewed

234 -

235 -

234 235 <PackageDisplay onClickPackage={handleInstall} packagesList={filteredPackages} setPackagesList={setFilteredPackages} loading={loading} />

252 + <div className="home-discover">

Stankov,Nikola N.M. @I453582 · 3 minutes ago

Maintainer

✓ 😊 💬 ✎ ⋮

Classes for different divs and tags. Looks good!

👍 1 😊

Reply...

Resolve thread

🔗

Start a new discussion...

253 + <h1 className="home-list-title">Discover</h1>

254 + <PackageDisplay onClickPackage={handleInstall} packagesList={filteredPackages} setPackagesList={setFilteredPackages} loading={loading} />

255 + </div>

256 + <div className="home-top-rated">

0 Assignees

None - assign yourself

Edit

0 Reviewers

None

Edit

Milestone

None

Edit

Time tracking

No estimate or time spent

?

Labels

None

Edit

Lock merge request

Unlocked

Edit

1 participant

Notifications

ⓧ

Reference: I453582/group2project...

Code review on my colleague Roel.

3. General code review

In this part I will review a part of the code I had to write unit test for. In this particular case we will review the code which is in the Users service of our backend RESTful API application. I will divide the review in 2 categories: **Code testability** and **Architecture**.

All of the reviews that follow have been discussed with the corresponding group members and decisions made as to whether the code should be left as it is due to current circumstances (time, difficulty, etc....) or to be refactored.

3.1. Code testability

We will take a look into some of the methods the Users service has:

- The 'Add user' method:

```
}

public boolean addUser(CreateUserRequestDTO request)
{
    if(userRepository.findUserByEmail(request.getEmail()) != null
        || !StringUtils.hasText(request.getEmail())
        || !StringUtils.hasText(request.getFirstName())
        || !StringUtils.hasText(request.getLastName())
        || !StringUtils.hasText(request.getPassword())
    )
    {
        return false;
    }
    String userID = UUID.randomUUID().toString();
    User user = User.builder()
        .id(userID)
        .email(request.getEmail())
        .password(request.getPassword())
        .firstName(request.getFirstName())
        .lastName(request.getLastName())
        .build();
    this.userRepository.save(user);
    return true;
}
```

As seen from the screenshot you can see that the code overall looks good and organized. That is true, however, the return method is Boolean. This makes the method really hard to test if the user has been added with the correct username, password, etc. There is no real value returned to compare to, so the only thing we can check is if the method returns true or false, which in my opinion is far from sufficient to determine if this method works correctly.

■ The 'Update user' method:

```
@Override
public void updateUser(UpdateUserRequestDTO request)
{
    Optional<User> user = userRepository.findById(request.getUserID());
    if(user.isEmpty() || !Objects.equals(user.get().getEmail(), request.getEmail())
        && userRepository.findUserByEmail(request.getEmail()) != null)
    {
        throw new InvalidUserException();
    }

    User updateUser = user.get();
    updateUser.setEmail(request.getEmail());
    updateUser.setFirstName(request.getFirstName());
    updateUser.setLastName(request.getLastName());

    userRepository.save(updateUser);
}

public List<UserDto> getAllUsers()
{
    return userRepository.findAll().stream().map(UserDtoConverter::convertToDto).toList();
}
```

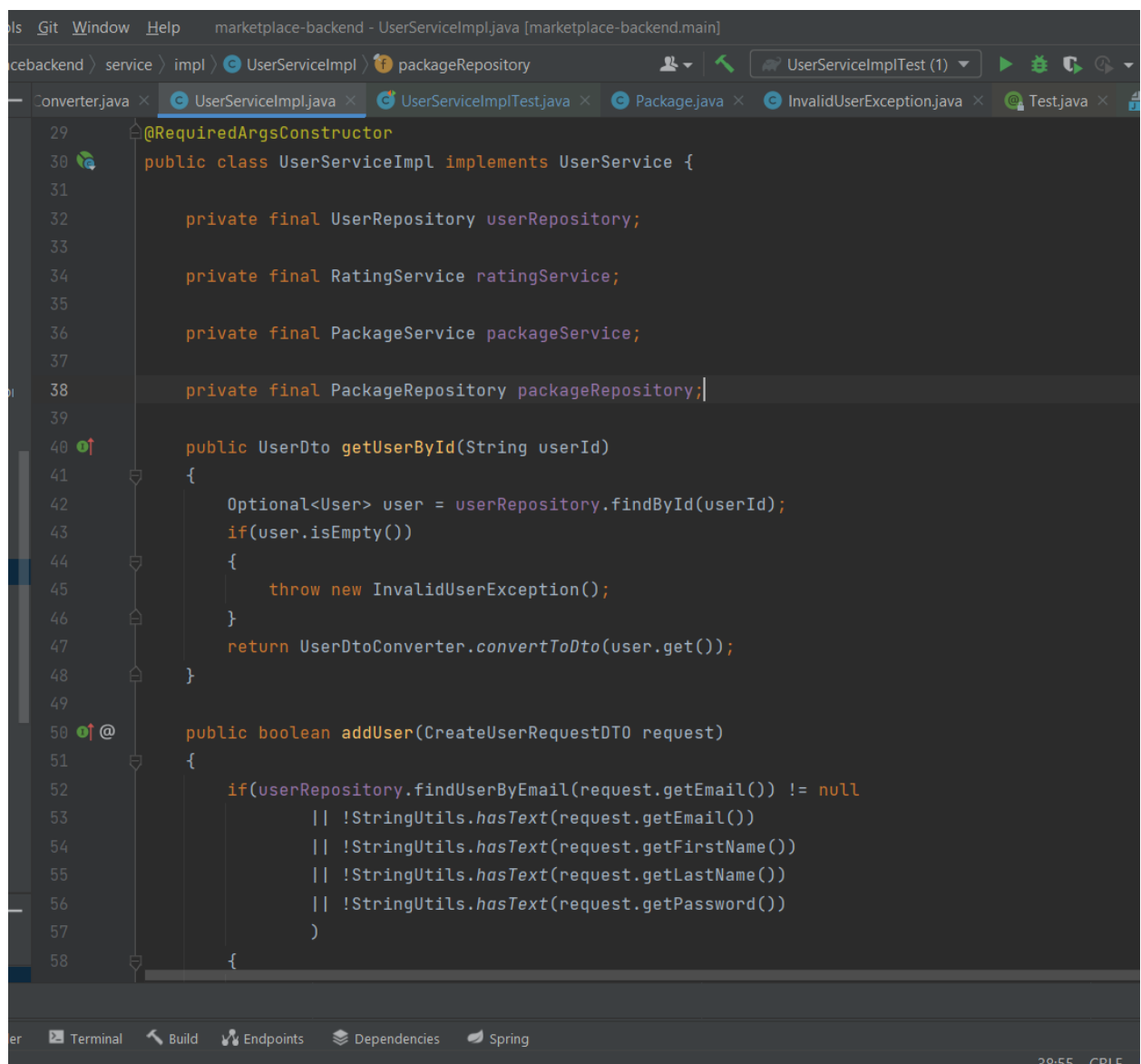
We have a similar thing going on in this method. The code itself is good, it validates the user properties as well and throws a custom exception which is handled in the front end. All that is great, but the method itself is a void method. This time it doesn't return anything so this is even harder to test.

3.2. Architecture

All in all, the architecture of the Ivanti marketplace project is pretty good. There is dependency inversion between different layers, ensured by different interfaces as well as dependency injections throughout the whole architecture. Of course, no one is perfect so there are some parts of the code I think can be improved in regards to following the SOLID principles.

*The things I am going to discuss right now have been discussed with my team members and decisions made. We will talk about the **Single Responsibility** principle from SOLID which by all means is a grey area and different people implement it differently. There is no 100% correct solution, the changes that I am going to suggest are solely based on my opinion.*

Let's look at the following code examples:



```
29  @RequiredArgsConstructor
30  public class UserServiceImpl implements UserService {
31
32      private final UserRepository userRepository;
33
34      private final RatingService ratingService;
35
36      private final PackageService packageService;
37
38      private final PackageRepository packageRepository;
39
40      public UserDto getUserById(String userId)
41      {
42          Optional<User> user = userRepository.findById(userId);
43          if(user.isEmpty())
44          {
45              throw new InvalidUserException();
46          }
47          return UserDtoConverter.convertToDto(user.get());
48      }
49
50      @
51      public boolean addUser(CreateUserRequestDTO request)
52      {
53          if(userRepository.findUserByEmail(request.getEmail()) != null
54             || !StringUtils.hasText(request.getEmail())
55             || !StringUtils.hasText(request.getFirstName())
56             || !StringUtils.hasText(request.getLastName())
57             || !StringUtils.hasText(request.getPassword())
58          )
59          {
```

```

public PackageDto getDownloadedPackagePerUser(String userID, String packageID) {
    Optional<User> user = userRepository.findById(userID);
    if(user.isEmpty()) {
        throw new InvalidUserException();
    }
    PackageDto packageDto = packageService.getPackageById(packageID);

    List<Package> downloadedPackages = user.get().getDownloadedPackages();
    if(downloadedPackages == null) {
        downloadedPackages = new ArrayList<>();
    }
    Package downloadedPackage = null;
    for(Package p : downloadedPackages) {
        if(p.getId().equals(packageDto.getId())) {
            downloadedPackage = p;
        }
    }
    if(downloadedPackage == null) {
        return null;
    }

    List<GetRatingDto> ratings = ratingService.getRatingsByPackage(packageID);

    return PackageDtoConverter.convertToDto(downloadedPackage, ratings);
}

```

```

public boolean removeDownloadedPackage(String userID, String packageID) {
    Optional<User> user = userRepository.findById(userID);
    if(user.isEmpty())
    {
        throw new InvalidUserException();
    }
    Optional<Package> packageModel = packageRepository.findById(packageID);
    if(packageModel.isEmpty())
    {
        throw new InvalidPackageException();
    }
    // Return false if the package is not downloaded
    if(getDownloadedPackagePerUser(user.get().getId(), packageModel.get().getId()) == null) {
        return false;
    }

    // Remove a rating for the package if the user had one
    Rating rating = ratingService.getRatingByRatingPackageAndUser(packageModel.get(), user.get());
    if(rating != null) {
        ratingService.deleteRating(rating.getId());
    }

    List<Package> downloadedPackages = user.get().getDownloadedPackages();
    if(downloadedPackages == null) {
        downloadedPackages = new ArrayList<>();
    }
    downloadedPackages.removeIf(dp -> dp.getId().equals(packageModel.get().getId()));
    user.get().setDownloadedPackages(downloadedPackages);
    userRepository.save(user.get());
    return true;
}

```

The code above is functional and seems to work fine. However, I think the clean architecture that we all have strived for in this project is being kind of broken in the shown case. We can see the User Service implementation class which is basically the implementation of the User Service interface in the pictures. I would argue that this service class does not follow the Single Responsibility principle (SRP) and I will explain why.

First of all, the SRP states that a class should only have one responsibility which of course is very vague and may be interpreted in different ways. That being said, I would argue that anything that has to do solely with users should be included in this class. However, we can see that apart from the 'User repository' class we also have the 'Package repository', 'Package service' and 'Ratings service'. The packages and ratings are definitely related to the user; however, I don't agree that they should be in the same class which purpose is to manage solely the user. I think, a separate service and hence controller to expose these joined endpoints (the methods shown in the pictures above) would make more sense and make the code cleaner and more readable.

What is more, we can see that the User Service depicted in the pictures also depend on other services. So, if something goes wrong with one of the services the whole User service will break which is not ideal as the User Service should solely manage the user independently of other components of the system.

4. Conclusion

To conclude, these were the point I wanted to pay attention to in this document. Of course, the code has been tested and reviewed in much more depth, but in this document I focused on the main things I wanted to pinpoint.