Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Інститут прикладного системного аналізу
Кафедра системного ароектування

Лабораторна робота №6

з курсу «*Проектування інформаційних систем*»

на тему: «Система автоматичного створення довідника користувача та оформлення коду за допомогою Coding Convention.»

Виконав:
студент групи ДА-71
Стефура Олег

Київ — 2020

Мета: за допомогою системи генерації довідника користувача створити документ у форматі PDF і HTML для архітектурної програмної моделі.

Задача:

1. Вивчити теги системи генерації керівництва користувача.
2. Створити опис для всіх класів API з описом призначення кожного класу, методів класу і членів класу.
3. Згенерувати документацію у форматах PDF, HTML.

Завдання:

1. Для кожного з класів API в код програми додати теги з описом керівництва користувача для архітектурної програмної моделі.
2. Обрати Coding Convention. Оформити код програми згідно Coding Convention.
2. Встановити налаштування системи автоматичного створення керівництва користувача.
3. Згенерувати HTLM документацію керівництва користувача.
4. Згенерувати PDF документацію керівництва користувача..

Хід роботи

Для мови програмування Java традиційним інструментом для генерації довідника користувача є Javadoc. Опишемо його основні теги:

/** */ - опис керівництва користувача включається в такі теги

@author – вказує автора коду

@version – вказує версію коду

@since – вказує, з якої версії доступний код

@see – посилання на інше місце в документації

@param – опис параметрів методу

@return – опис значень, що повертає метод

@exception ex_name - опис виключення, яке генерується в методі

Застосуємо вказані теги для опису головного класу API програми.

Опишемо сам клас, поля класу:

```java
/**
 * This class is used for handling outer requests
 *
 * @author oleg-stefura
 * @version v1
 * */
@RestController
@RequestMapping("/v1/users")
public class UsersController {
    /** Is used for manipulating with User models*/
    @Autowired
    private UserService userService;

    /** Is used for manipulating with Goal models*/
    @Autowired
    private GoalService goalService;
```

Рисунок 6.1 – Опис класу та полів

Опишемо методи.

```java
/**
 * Handles request on main URL (/users) and used to get the list of users.
 * The response is auto mapped from UserDto to JSON
 *
 * @param email optional parameter for searching user by email. If null - all users will be send
 * @return list of all users or users with particular email if email-param is not null
 * @see UserDto
 * */
@GetMapping
@ResponseBody
public List<UserDto> getAllUsers(@RequestParam(required = false) String email) {
    if (email == null) {
        return userService.getAllUsers().stream().map(MappingService::mapUserToDto).collect(Collectors.toList());
    }

    return Arrays.asList(mapUserToDto(userService.getUserByEmail(email)));
}
```

Рисунок 6.2 – Метод для отримання всіх користувачів

```java
/**
 * Handles request on URL (/users/{id}) and used to get the particular user.
 * The response is auto mapped from UserDto to JSON
 *
 * @param id mandatory parameter for searching user by particular id
 * @return particular user by ID
 * @exception com.delta.GoalTracker.exceptions.UserNotFoundException is thrown when user with particular id s not found
 * @see UserDto
 * */
@GetMapping("/{id}")
@ResponseBody
public UserDto getUserById(@PathVariable Integer id) {
    return mapUserToDto(userService.getUserById(id));
}
```

Рисунок 6.3 – Метод для отримання користувача за ІД

```java
    /**
     * Handles POST request on URL (/users) and used to create a new user.
     * The response is newly created user and is auto mapped from UserDto to JSON
     *
     * @param userToCreate Info about user to create
     * @return created user
     * @see UserDto
     * */
    @PostMapping
    @ResponseBody
    public UserDto createUser(@RequestBody UserDto userToCreate) {
        User createdUser  = mapDtoToUser(userToCreate);
        createdUser = userService.addUser(createdUser);

        return mapUserToDto(createdUser);
    }
```

Рисунок 6.4 – Метод для створення нового користувача

```java
    /**
     * Handles GET request on URL (/users/{id}/goals) and returns the list of goals of particular user.
     * If status param is not null - only the goals with particular status will be returned
     * The response is auto mapped from GoalDto to JSON
     *
     * @param userId id of the particular user
     * @param status status for goals to be searched
     * @return found goals or empty list is there are not any goals
     * @see GoalDto
     * */
    @GetMapping("/{userId}/goals")
    @ResponseBody
    public List<GoalDto> getUserGoals(@PathVariable Integer userId, @RequestParam(required = false) String status) {
        if (status == null) {
            return goalService.getUserGoals(userService.getUserById(userId))
                    .stream().map(MappingService::mapGoalToDto).collect(Collectors.toList());
        }

        return goalService.getUserGoalsWithStatus(userService.getUserById(userId), status)
                .stream().map(MappingService::mapGoalToDto).collect(Collectors.toList());
    }
```

Рисунок 6.5 – Метод для отримання цілей користувача

Продовжимо так далі.

Наступний крок – додамо нову залежність у проект для генерації документації, а саме додамо maven-plugins.

```xml
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-javadoc-plugin</artifactId>
                <version>3.2.0</version>
            </plugin>
```

Рисунок 6.6 – Плагін для генерації документації

Запустимо команду maven для генерації документації:

```
\Labs\lab4\GoalTracker>mvn javadoc:javadoc
```

Рисунок 6.7 – Команда для генерації документації

```
Generating C:\Users\stefu\Desktop\Source\Studying\Current course\info_system_design\Labs\lab4\GoalTracker\target\site\apidocs\index.html...
Generating C:\Users\stefu\Desktop\Source\Studying\Current course\info_system_design\Labs\lab4\GoalTracker\target\site\apidocs\overview-summary.html...
Generating C:\Users\stefu\Desktop\Source\Studying\Current course\info_system_design\Labs\lab4\GoalTracker\target\site\apidocs\help-doc.html...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  30.090 s
[INFO] Finished at: 2020-12-28T11:31:31+02:00
[INFO] ------------------------------------------------------------------------
```

Рисунок 6.8 – Успішна генерація

Переглянемо отримані результати:

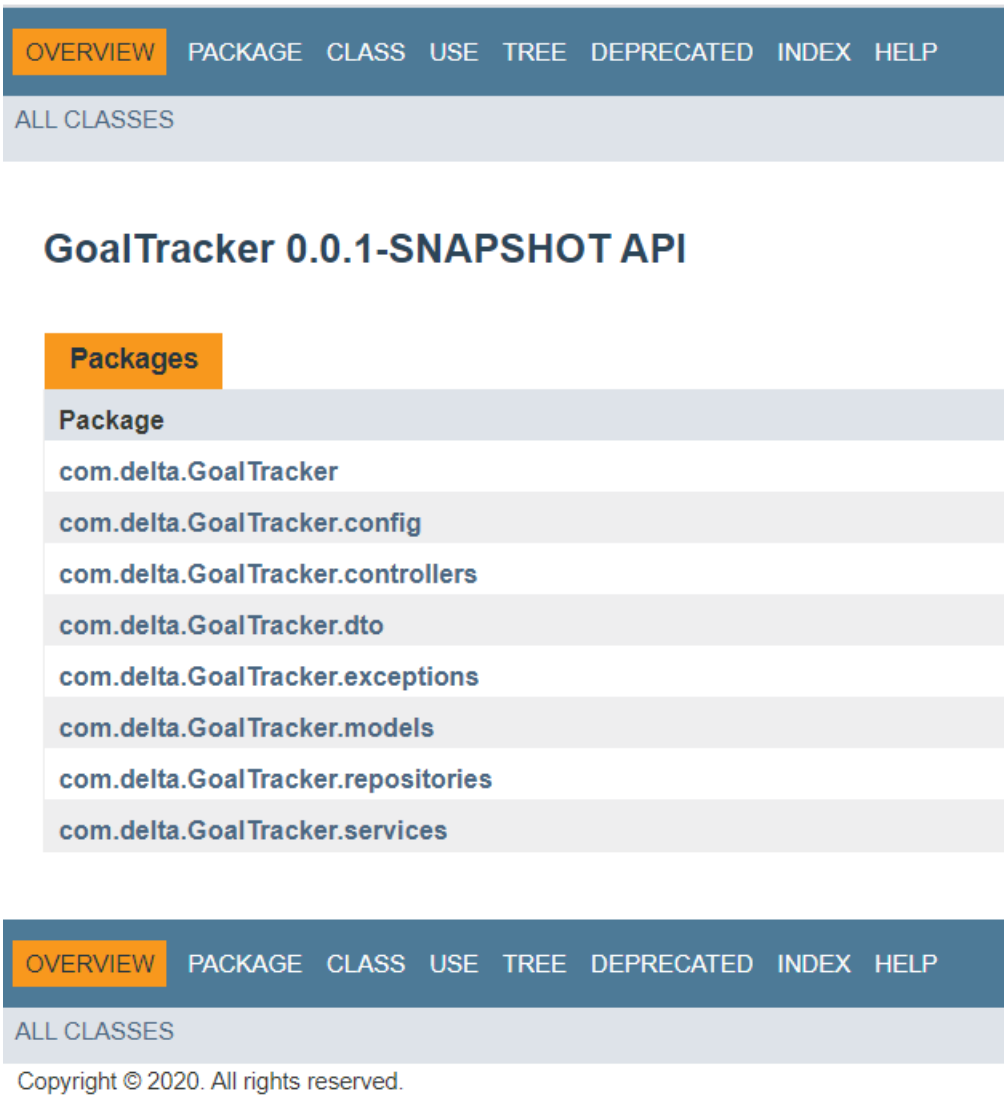


Рисунок 6.9 – Головна сторінка

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package com.delta.GoalTracker.controllers

Class UsersController

java.lang.Object
    com.delta.GoalTracker.controllers.UsersController

```
@RestController
@RequestMapping("/v1/users")
public class UsersController
extends Object
```

This class is used for handling outer requests

Version:

v1

Author:

oleg-stefura

Constructor Summary

Constructors

Constructor
`UsersController()`

Рисунок 6.10 – Загальний опис класу

getAllUsers

```
@GetMapping @ResponseBody public List<UserDto> getAllUsers(@RequestParam(required=false) String email)
```

Handles request on main URL (/users) and used to get the list of users. The response is auto mapped from UserDto to JSON

Parameters:

`email` - optional parameter for searching user by email. If null - all users will be send

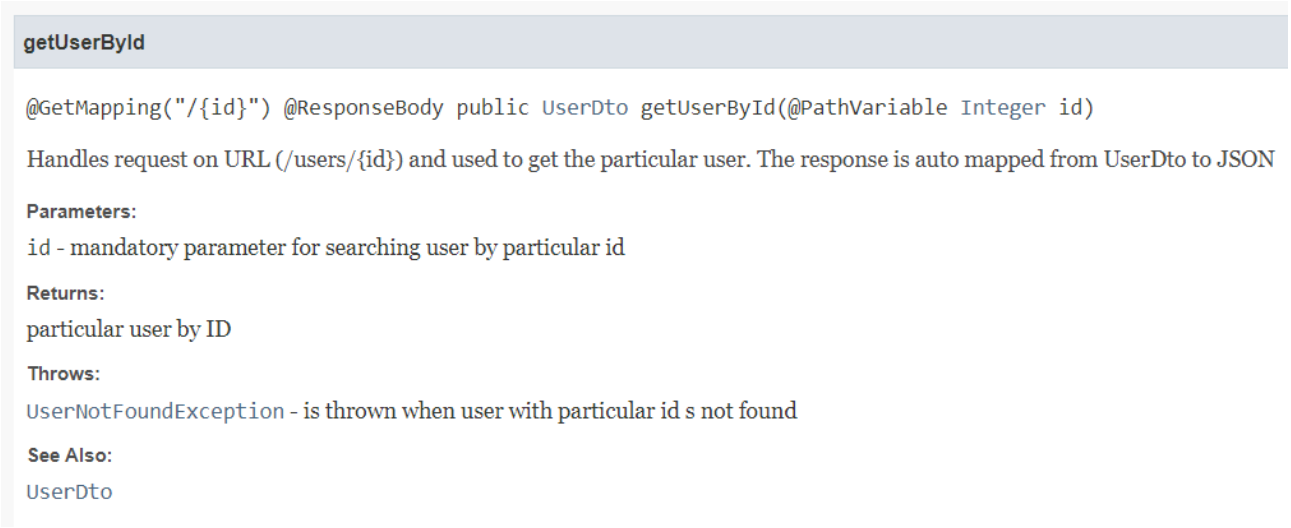
Returns:

list of all users or users with particular email if email-param is not null

See Also:

`UserDto`

Рисунок 6.11 – Метод для пошуку всіх користувачів



getUserById

```
@GetMapping("/{id}") @ResponseBody public UserDto getUserById(@PathVariable Integer id)
```

Handles request on URL (/users/{id}) and used to get the particular user. The response is auto mapped from UserDto to JSON

Parameters:

`id` - mandatory parameter for searching user by particular id

Returns:

particular user by ID

Throws:

`UserNotFoundException` - is thrown when user with particular id s not found

See Also:

`UserDto`

Рисунок 6.12 – Пошук користувача за ІД

getUserGoals

```
@GetMapping("/{userId}/goals") @ResponseBody public List<GoalDto> getUserGoals(@PathVariable Integer userId, @RequestParam(required=false) String status)
```

Handles GET request on URL (/users/{id}/goals) and returns the list of goals of particular user. If status param is not null - only the goals with particular status will be returned The response is auto mapped from GoalDto to JSON

Parameters:

`userId` - id of the particular user

`status` - status for goals to be searched

Returns:

found goals or empty list is there are not any goals

See Also:

`GoalDto`

Рисунок 6.13 – Пошук цілей користувача

Тепер згенеруємо документацію в PDF-форматі. Для цього використаємо утиліту HTMLDOC (https://www.msweet.org/htmldoc/index.html). Завантажимо та відкриємо програму. Виберемо тип документа «Web Page», додамо файл документації нашого класу.

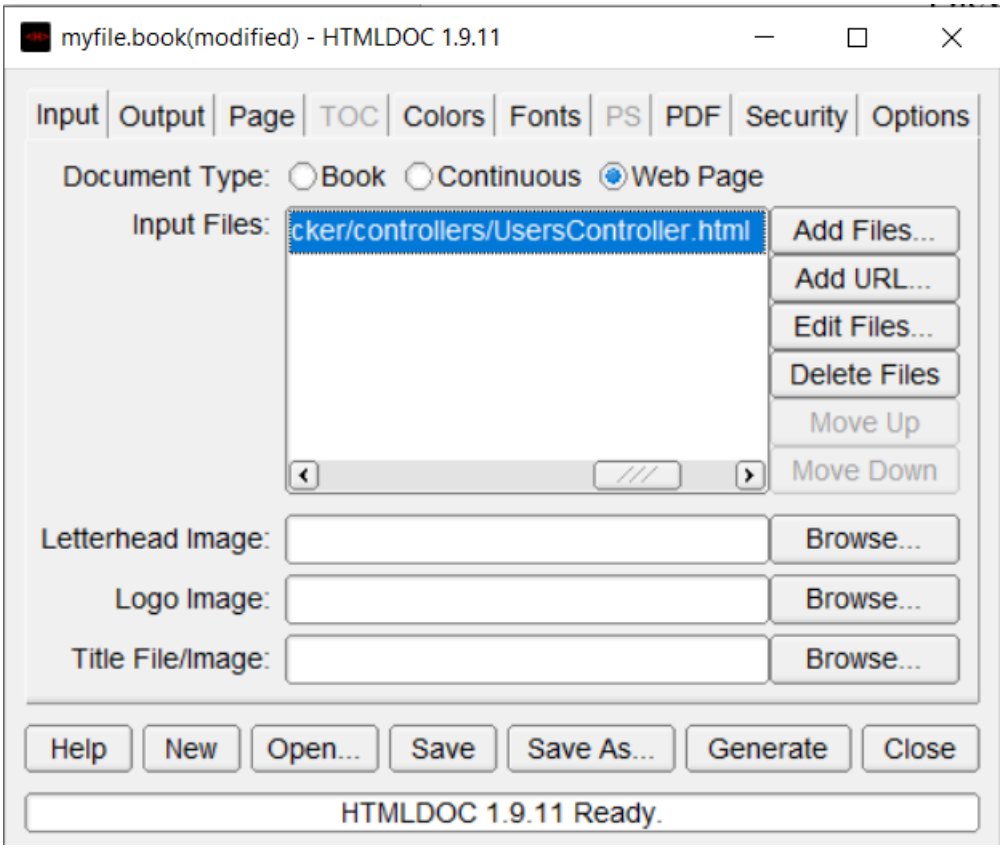


Рисунок 6.14 – Стартове вікно програми HTMLDOC

У вкладці «Output» вкажемо тип PDF, ім’я файлу та натиснемо кнопку «Generate».

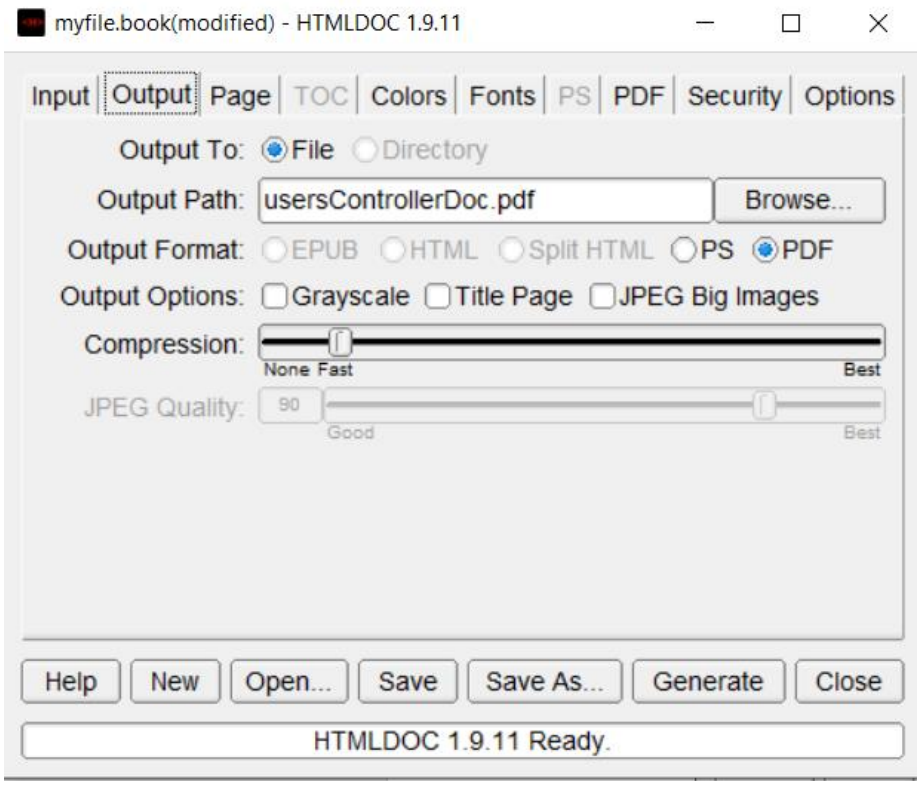
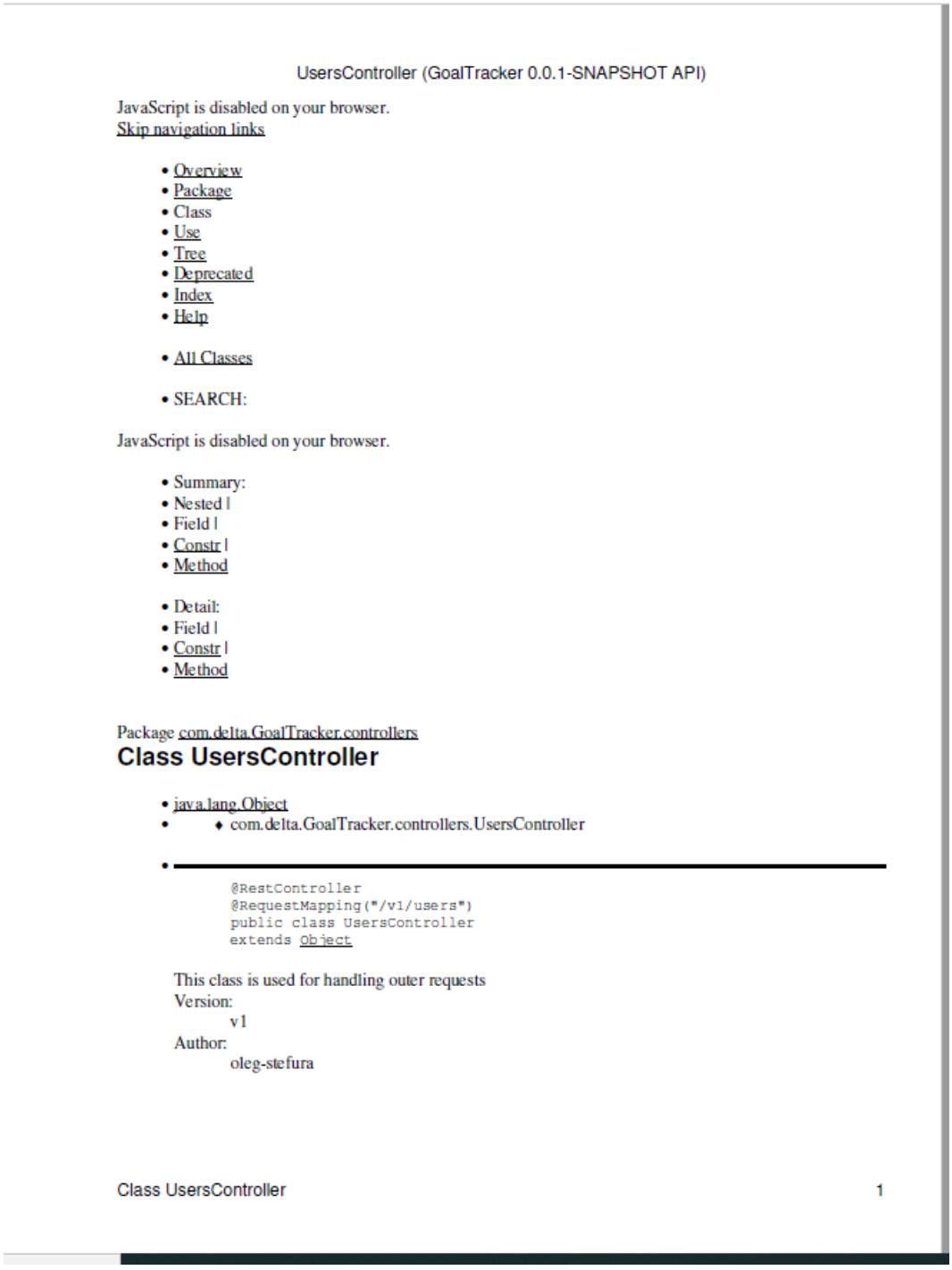


Рисунок 6.15 – Генерація PDF-документу

Відкриємо згенерований документ:



UsersController (GoalTracker 0.0.1-SNAPSHOT API)

JavaScript is disabled on your browser.
Skip navigation links

- Overview
- Package
- Class
- Use
- Tree
- Deprecated
- Index
- Help

- All Classes

- SEARCH:

JavaScript is disabled on your browser.

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

Package com.delta.GoalTracker.controllers

Class UsersController

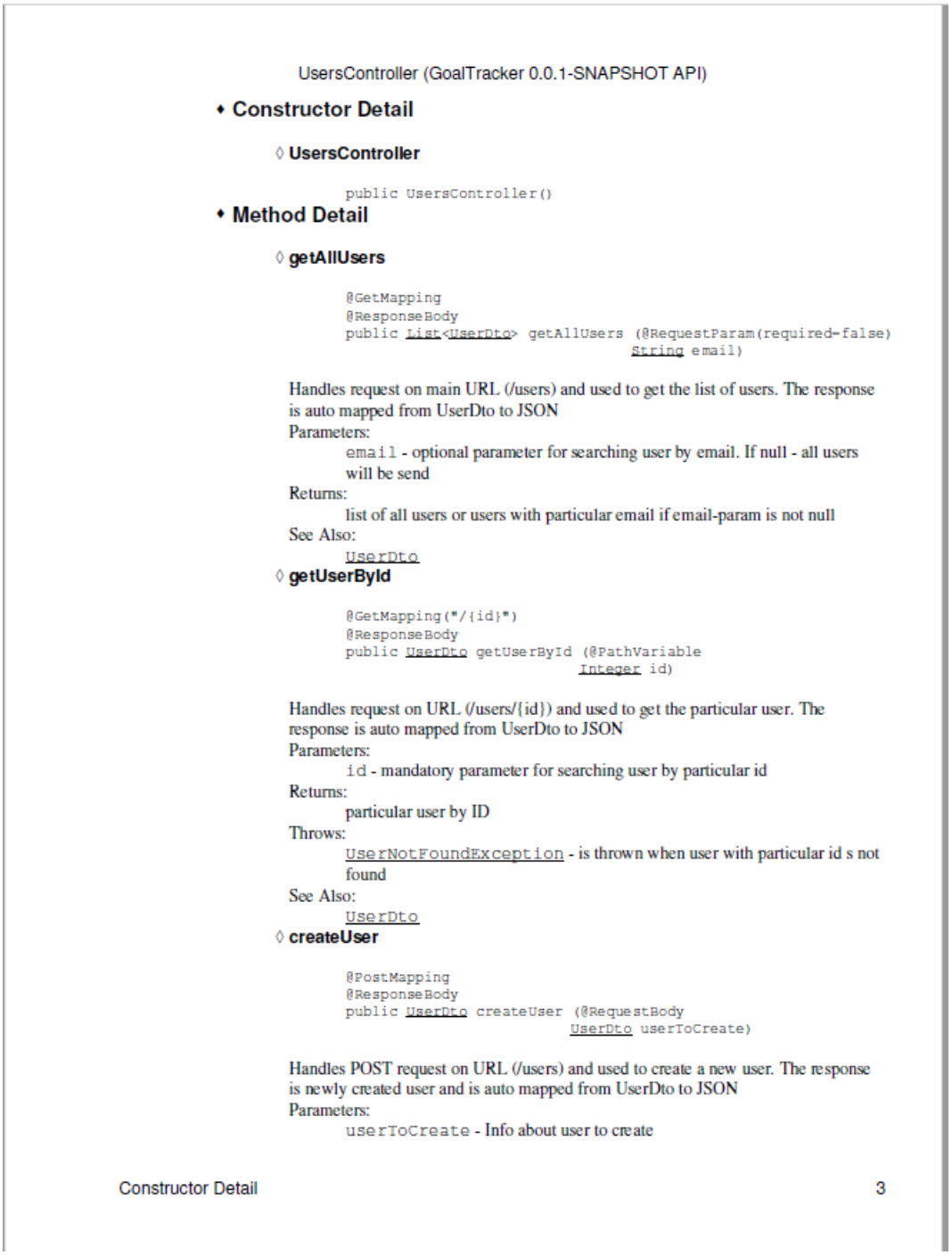
- java.lang.Object
- ♦ com.delta.GoalTracker.controllers.UsersController

```
@RestController
@RequestMapping("/v1/users")
public class UsersController
extends Object
```

This class is used for handling outer requests
Version:
v1
Author:
oleg-stefura

Class UsersController 1

Рисунок 6.16 – Згенерований документ (головна сторінка)



UsersController (GoalTracker 0.0.1-SNAPSHOT API)

♦ **Constructor Detail**

◊ **UsersController**

```
public UsersController()
```

♦ **Method Detail**

◊ **getAllUsers**

```
@GetMapping
@ResponseBody
public List<UserDto> getAllUsers (@RequestParam(required=false)
                                  String email)
```

Handles request on main URL (/users) and used to get the list of users. The response is auto mapped from UserDto to JSON
Parameters:
email - optional parameter for searching user by email. If null - all users will be send
Returns:
list of all users or users with particular email if email-param is not null
See Also:
UserDto

◊ **getUserById**

```
@GetMapping("/{id}")
@ResponseBody
public UserDto getUserById (@PathVariable
                            Integer id)
```

Handles request on URL (/users/{id}) and used to get the particular user. The response is auto mapped from UserDto to JSON
Parameters:
id - mandatory parameter for searching user by particular id
Returns:
particular user by ID
Throws:
UserNotFoundException - is thrown when user with particular id s not found
See Also:
UserDto

◊ **createUser**

```
@PostMapping
@ResponseBody
public UserDto createUser (@RequestBody
                           UserDto userToCreate)
```

Handles POST request on URL (/users) and used to create a new user. The response is newly created user and is auto mapped from UserDto to JSON
Parameters:
userToCreate - Info about user to create

Constructor Detail 3

Рисунок 6.17 – Згенерований документ (3-тя сторінка)

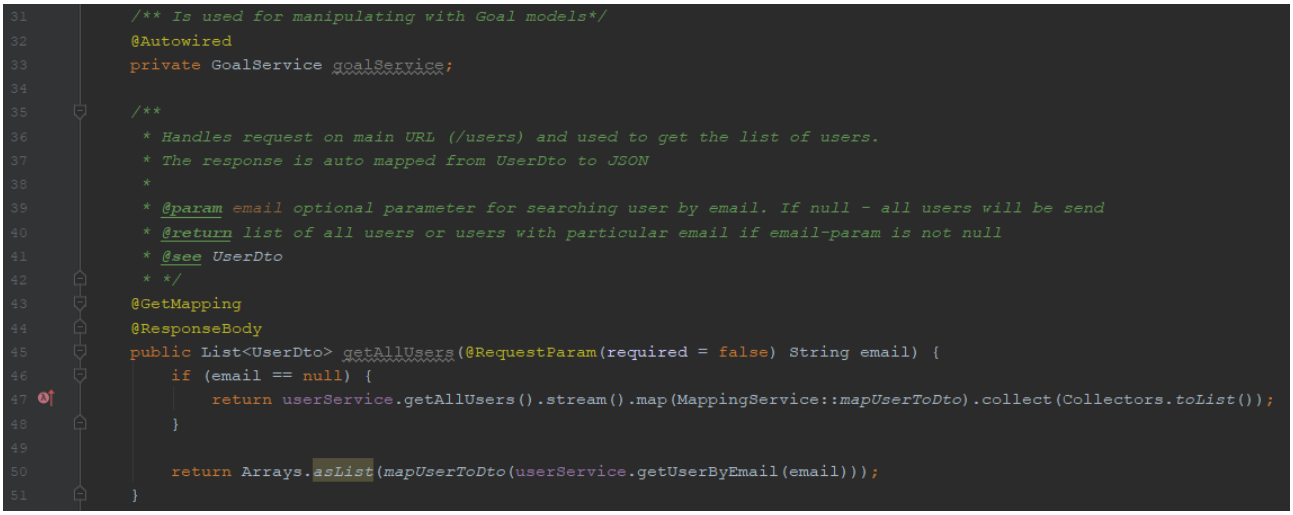
Варто зазначити, що код програми написаний за найбільш поширеним Java Code Convention. Зокрема, одними із важливих його пложень є наступні:

- декларація кожного класу проводиться в окремому файлі. Структура файлу наступна: назва пакету, статичний імпорт, не-статичний імпорт, назва класу, тіло класу (рис.6.18)
- всі частини документу розділяються одним порожнім рядком (рис.6.18)
- назва класу – строго із великої літери, CamelCase (рис. 6.18)

- назва пекету – строго в нижньому регістрі (рис.6.18)
- назви змінних та методів – CamelCase, починаючи з малої літери (рис.6.19)
- формат методу наступний: <access> [<static>] <return type> <name, camelCase>([ParamType1 paramName1], [ParamType2 paramName2], …) { } (всі вкзаані елменети розділяються одним пробілом, окрім дужок з параметрами, які йдуть одразу після назви методу; після символу “{” йде перехід на новий рядок); (рис.6.19)
- допускається тільки один вираз на рядок; (рис.6.19)
- інше

```
1   package com.delta.GoalTracker.controllers;
2
3   import static com.delta.GoalTracker.services.MappingService.*;
4
5   import com.delta.GoalTracker.dto.GoalDto;
6   import com.delta.GoalTracker.dto.UserDto;
7   import com.delta.GoalTracker.models.Goal;
8   import com.delta.GoalTracker.models.User;
9   import com.delta.GoalTracker.services.GoalService;
10  import com.delta.GoalTracker.services.MappingService;
11  import com.delta.GoalTracker.services.UserService;
12  import org.springframework.beans.factory.annotation.Autowired;
13  import org.springframework.web.bind.annotation.*;
14  import java.util.Arrays;
15  import java.util.List;
16  import java.util.stream.Collectors;
17
18  /**
19   * This class is used for handling outer requests
20   *
21   * @author oleg-stefura
22   * @version v1
23   * */
24  @RestController
25  @RequestMapping("/v1/users")
26  public class UsersController {
27      /** Is used for manipulating with User models*/
28      ...
159 }
```

Рисунок 6.18



```java
    /** Is used for manipulating with Goal models*/
    @Autowired
    private GoalService goalService;

    /**
     * Handles request on main URL (/users) and used to get the list of users.
     * The response is auto mapped from UserDto to JSON
     *
     * @param email optional parameter for searching user by email. If null - all users will be send
     * @return list of all users or users with particular email if email-param is not null
     * @see UserDto
     * */
    @GetMapping
    @ResponseBody
    public List<UserDto> getAllUsers(@RequestParam(required = false) String email) {
        if (email == null) {
            return userService.getAllUsers().stream().map(MappingService::mapUserToDto).collect(Collectors.toList());
        }

        return Arrays.asList(mapUserToDto(userService.getUserByEmail(email)));
    }
```

Рисунок 6.19

Висновки: в результаті виконання роботи було успішно згенеровано документацію користувача, продемонстровано основні теги інстурменту для генерації для МП Java – Javadoc, у поєднанні із системою автоматичного збирання Maven. Показаний процес демонструє надзвичайну легкість, а результат – високу ефективність.

Результати представлені у вигляді HTML сторінок, але за бажанням, за допомогою додаткової утиліти, можна легко згенерувати PDF документацію із поередньо створеної HTML-документації.

Вкінці приведено деякі основні положення із Java Code Convention, дотримання яких при написанні коду значно спрощує взаємодію між програмістами, читання чужого коду.