

**GETİR**

**Technical Specification**

|  |  |
| --- | --- |
| **Title** |  |
| ***Document Name*** | **Reading Is Good Technical Design** |
| ***Version No.*** | 1.0 |
| ***Date*** | 16.05.2022 |
| ***Document Author(s)*** | Sevim Demir |

Version History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change** | **Sections Affected** |
| ***1.0*** | 16.05.2022 | Sevim Demir | Create | All |
|  |  |  |  |  |

**Table of Contents**

[1. Introduction 1](#_Toc103693279)

[Business Requirement 1](#_Toc103693280)

[1.1.1. Functional 1](#_Toc103693281)

[1.1.2. Non-Functional 1](#_Toc103693282)

[1.1.2.1. Timings & Frequency 1](#_Toc103693283)

[1.1.2.2. Security 1](#_Toc103693284)

[2. Technical Design 2](#_Toc103693285)

[Technical Approach 2](#_Toc103693286)

[3. Design Elements 3](#_Toc103693287)

[Db Objects 3](#_Toc103693288)

[3.1.1. Users Object 3](#_Toc103693289)

[3.1.2. Books Object 3](#_Toc103693290)

[3.1.3. Orders Object 3](#_Toc103693291)

[3.1.4. Statistics Object 3](#_Toc103693292)

[Public APIs 4](#_Toc103693293)

[3.2.1 User Controller 4](#_Toc103693294)

[3.2.2 Book Controller 5](#_Toc103693295)

[3.2.3 Order Controller 6](#_Toc103693296)

[3.2.4 Statistics Controller 7](#_Toc103693297)

[4. Testing 7](#_Toc103693298)

[5. Requirements 7](#_Toc103693299)

[5.1 Technical Requirements 7](#_Toc103693300)

[6. Setup&Run 8](#_Toc103693301)

# Introduction

ReadingIsGood is an online books retail firm which operates only on the Internet. Main target of ReadingIsGood is to deliver books from its one centralized warehouse to their customers within the same day. That is why stock consistency is the first priority for their vision operations.

## Business Requirement

### Functional

• Query Monthly Statistics

• Registering New Customer

• Placing a new order

• Tracking the stock of books

• List all orders of the customer

• Viewing the order details

### Non-Functional

### Timings & Frequency

System should be consistent in terms of book stocks

System should be atomic in terms of book stocks

### Security

Bearer token authentication and secure APIs

### 

# Technical Design

## Technical Approach

This is the backend server or online book selling application. Developed by using SpringBoot(2.6.7) and MongoDB. It runs on an embedded Tomcat via port 8080.

<http://localhost:8080/>

Server runs with Spring-Security using Bearer Token authentication. Email will be used as user\_name

# Design Elements

## Db Objects

### Users Object

|  |  |
| --- | --- |
| **Column Name** | **Type** |
| id | String |
| firstName | String |
| lastName | String |
| email | String |

### Books Object

|  |  |
| --- | --- |
| **Column Name** | **Type** |
| id | String |
| name | String |
| desc | String |
| price | number |
| stock | number |

### Orders Object

|  |  |
| --- | --- |
| **Column Name** | **Type** |
| id | String |
| name | String |
| desc | String |
| price | number |
| stock | number |

### Statistics Object

|  |  |
| --- | --- |
| **Column Name** | **Type** |
| id | String |
| orderCount | number |
| revenue | number |
| soldBookCount | number |
| month | number |
| year | number |

## Public APIs

### User Controller

#### Login to the system

**POST** /api/v1/user/login

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| email | string | **Required**. email of user. |
| password | string | **Required**. password of user. |

#### Create New User

**POST** /api/v1/user/new

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| email | string | **Required**. email of user. |
| firstName | string | **Required**. first name of the user. |
| lastName | string | **Required**. last name of the user. |
| password | string | **Required**. password of user. |

#### Displays the Orders created by user with paging

**GET** /api/v1/user/order

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| page | number | **Required**. page index of orders |
| size | number | **Required**. page window size of the orders. |

### Book Controller

#### Getting single book with id

**GET** /api/v1/book/{id}

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| id | String | **Required**. id of the book. |

#### Listing all books

**GET** /api/v1/book/all

#### Create Single Book

**POST** /api/v1/ book /new

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| name | string | **Required**. name of the book. |
| desc | string | **Required**. Description of the book. |
| price | number | **Required**. price of the book. |
| stock | number | **Required**. available number of book. |

#### Updates Single Book

**PUT** /api/v1/book/update

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| İd | String | **Required**. id of the book. |
| name | string | **Required**. name of the book. |
| desc | string | **Required**. Description of the book. |
| price | number | **Required**. price of the book. |
| stock | number | **Required**. available number of book. |

### Order Controller

#### Gets single order with specific ID

**GET** /api/v1/order/{id}

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| İd | String | **Required**. id of order. |

#### Listing all orders of the user

**GET** /api/v1/order/all

#### List all order between given date

**GET** /api/v1/order/list

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| from | date | **Required**. start date.(ISO formatted) |
| to | date | **Required**. end date.(ISO formatted) |

#### Creates new order

**POST** /api/v1/order/new

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| bookId | string | **Required**. id of sold book |
| quantity | number | **Required**. number of book sold |

### Statistics Controller

#### Gets montly statistic

**GET** /api/v1/stat/monthly

# Testing

|  |  |
| --- | --- |
| **Service** | **Coverage (%)** |
| UserServiceTest | 61% |
| BookServiceTest | 63% |
| OrderServiceTest | 83% |
| StatisticServiceTest | 100% |

# Requirements

### 5.1 Technical Requirements

* JDK 11
* Spring boot
* Spring Security with JWT
* MongoDB
* Docker
* Lombok, mapstruct, swagger, mockito, faker

# Setup&Run

This project uses Docker as container. In order to run project you need to pre-install Docker.

<https://docs.docker.com/get-docker/>

Before running project install Java 11+ and set JAVA\_HOME as enviroment variable.

./mvnw clean install

docker-compose build --no-cache

docker-compose up --force-recreate

After running project you can visit <http://localhost:8080/swagger-ui/index.html> for open API specification.